Making Everything Easier!™

Visual Studio® 2010

ALL-IN-ONE

FOR DUMMIES®

7 BOOKS IN 1

- Visual Studio 2010 Overview
- Getting Started
- Coding
- Building Applications
- Data Access
- Going the Extra Mile
- Extending the Family

Andrew Moore

Visual Studio® 2010 ALL-IN-ONE FOR DUMMIES®

by Andrew Moore

WILEY

Wiley Publishing, Inc.

Visual Studio® 2010 All-in-One For Dummies®

Published by
Wiley Publishing, Inc.
111 River Street
Hoboken, NJ 07030-5774

www.wiley.com

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002.

For technical support, please visit www.wiley.com/techsupport.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Control Number: 2010928464

ISBN: 978-0-470-53943-9

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgments

We're proud of this book; please send us your comments at http://dummies.custhelp.com. For other comments, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002.

Some of the people who helped bring this book to market include the following:

Acquisitions, Editorial, and Media Development

Project Editor: Kelly Ewing

Acquisitions Editor: Katie Feltman

Technical Editor: Russ Mullen

Editorial Manager: Jodi Jensen

Media Development Project Manager: Laura Moss-Hollister

Media Development Assistant Project Manager: Jenny Swisher

Editorial Assistant: Amanda Graham

Sr. Editorial Assistant: Cherie Case

Cartoons: Rich Tennant (www.the5thwave.com)

Composition Services

Project Coordinator: Katherine Crocker

Layout and Graphics: Carrie A. Cesavice, Ashley Chamberlain, Samantha K. Cherolis

Proofreaders: Rebecca Denoncour, Evelyn Wellborn

Indexer: Becky Hornyak

Publishing and Editorial for Technology Dummies

Richard Swadley, Vice President and Executive Group Publisher

Andy Cummings, Vice President and Publisher

Mary Bednarek, Executive Acquisitions Director

Mary C. Corder, Editorial Director

Publishing for Consumer Dummies

Diane Graves Steele, Vice President and Publisher

Composition Services

Debbie Stailey, Director of Composition Services

Contents at a Glance

Table of Contents

Book II: Getting Started with Visual Studio 79

Chapter 1: Installing Visual Studio Professional 81

Chapter 2: Browsing Visual Studio 91

Chapter 3: Making Connections with Visual Studio 105

Introduction

Visual Studio 2010 is more than just the next version of Visual Studio to use with the next version of the .NET Framework. Visual Studio 2010 continues Microsoft's attempt to position Visual Studio as a tool you can use for the upstream and downstream development activities that sandwich actual code writing. For example, you can use Visual Studio to visually model the entities you want to create in code. Unlike other modeling tools that have only a one-way relationship between the model and the code generation, your code stays synchronized with the model.

Visual Studio 2010 is a complete rewrite of the user interface, but don't worry; the familiar commands and tools that you have come to know and love are still there. You'll notice a completely revised, intuitive, and modern user interface that showcases the tremendous leaps that Visual Studio has made since the release of Visual Studio .NET back in 2002.

Visual Studio 2010 provides a dizzying array of editors, controls, designers, and supporting tools for developing software. Getting mired in the details of using these tools is a productivity killer. This book uses plain English to show you how to use Visual Studio 2010 to get busy building software while ignoring unnecessary details. Use this book to focus on the work that pays the bills and to

- Improve your individual efficiency and productivity as a developer.
- Display proficiency in selecting the right Visual Studio 2010 tools required to develop a solution.
- Employ Visual Studio 2010 to guide and improve your individual software-development practices or those of a team.
- Navigate the many project types, editors, and designers in Visual Studio 2010.
- Increase your confidence and professionalism in using the development environment of choice for developers of solutions based on the Microsoft platform.
- Determine the role of Visual Studio in your software development process, whether you're working solo or on a team of 20.

Who Should Read This Book?

A little something for everyone is in this book, whether you're brand-new to software development or an old pro. No matter what your skill level is, you need this book if you see yourself in any of these categories:

- **New application developers:** Whether you're a student, a graduate who just landed your first programming job, or a power user looking to expand your horizons, you'll find everything you need to get productive with Visual Studio 2010 Professional.
- **Existing .NET developers:** Not sure when you can make the jump to version 4 of the .NET Framework? Never fear. You can start using Visual Studio 2010 right now, with previous versions of the .NET Framework. I show you how, in Book II, Chapter 4. Plus, see how to convert your existing applications and use what you already know to get productive.
- **Existing developers using other technologies:** Whether you're coming from Visual Basic 6 or Java, you'll find lots of no-frills examples to help you get started with Visual Studio 2010.
- **Advanced developers on the bleeding edge:** Even if you've used the .NET Framework since it was in beta, this book shows you how to use Visual Studio 2010 for the latest guidance and best practices in software development.

Visual Studio 2010 isn't just for developers anymore. Increasingly, software is developed as part of a team effort. Visual Studio 2010 has increased its scope to encompass more aspects of the software development life cycle. As a result, all roles on the team are increasingly affected. Regardless of your role, you may find this book helpful if you fill any of these roles:

- **Managers, leads, and supervisors** need to understand the productivity gains and best-practices guidance offered by Visual Studio 2010. These gains can be useful for improving team processes, as well as for evaluating programmer productivity.
- **Architects, designers, and analysts** will find new tools designed to improve collaboration among analysis, design, and implementation steps.
- **Developers, testers, and technologists** use Visual Studio 2010 to develop and test software. As such, this integral part of the software development process requires you to know how to harness its many features into a set of steps that supports a specific project's development requirements.

About This Book

In this book, you find out how to use Visual Studio 2010 Professional Edition to build these types of applications:

- Windows applications
- Web sites and Web services
- Mobile applications
- Native applications using C++

You may be surprised that Visual Studio 2010 has moved way beyond building merely traditional applications. You can use it to build and manage solutions for SQL Server databases, SharePoint sites, Windows Workflow applications, BizTalk packages, and many other enterprise server solutions. I discuss most of these topics throughout the book.

The message of this book is mostly how to use Visual Studio to improve your personal productivity as a developer, whether you're working solo or as part of a team.

The book focuses primarily on Visual Studio 2010 Professional Edition, although many examples work in other editions, too. Many developers, regardless of the size of their shops, use Visual Studio 2010 Professional Edition.

You can't talk about Visual Studio 2010 without also covering the .NET Framework. This book covers the .NET Framework at a very high level and in the context of demonstrating the features of Visual Studio 2010.

The book is mostly "language agnostic," although (just like in real life) the language best suited for the job is used to demonstrate the material. In most Microsoft shops, there is a preference for either Visual Basic or C#. For that reason, many chapters use Visual Basic examples when they could have just as easily used C# examples.

Despite the preceding statement, this book isn't a learn-to-program book or a language-syntax book. If you're new to programming, consider checking out a beginning programming book or course. If you're a hobbyist or a new programmer, you may find all the tools in Visual Studio 2010 to be overwhelming or outside the bounds of your budget. In that case, consider using a Visual Studio Express Edition, which is free.

Foolish Assumptions

You'll get the most out of this book if you already know how to use basic programming constructs, such as `for` loops and `if...then` statements. Even though this book doesn't teach you how to program, it does share guidance and tips on the use of best practices. Even if you've never programmed, you can still use the examples in this book to start creating basic Windows and Web applications using Visual Studio 2010.

Here are some other assumptions this book makes about you:

- **You have little or no experience with object-oriented programming (OOP).** Becoming an OOP whiz takes many years of hands-on practice. This book can help lay the groundwork for your OOP training and show you the OOP features in Visual Studio 2010. Book III introduces you to OOP.
- **You have little or no experience in using Visual Studio or the .NET Framework.** If you have plenty of experience with Visual Studio or the .NET Framework, you can reuse that knowledge with this version of Visual Studio. Either way, you are walked through all the examples, step by step.
- **You don't have formal training in computer science.** This book offers technical explanations of what Visual Studio is doing behind the scenes when it's relevant to helping you understand the topic.

Conventions Used in This Book

This book uses a helpful set of conventions to indicate what needs to be done or what you see on-screen.

Stuff you type

When you are asked to type something, like a command or an entry in a text box, the text looks like this:

```
Type me
```

Menu commands

When you are given a specific set of menu commands to use, they appear in this format:

File➪New➪Web Site

In this example, you should click the File menu, choose the New menu item, and then choose the Web Site menu item.

Display messages

If a specific message is referred to that you see on your screen, it looks like this on the page:

```
This is a message displayed by an application.
```

All code in the book also looks like this.

How This Book Is Organized

This book is organized so that you don't have to read it from cover to cover. To get the most out of the book, use the Table of Contents or Index to find specific topics. The seven mini-books cluster common tasks for which you might use Visual Studio 2010 to develop software. This section provides a brief overview of what you can find in each small book.

Book I: Visual Studio 2010 Overview

Book I is a good place to start if you're new to Visual Studio or .NET, or just want a refresher. In Book I, you get the lowdown on using Visual Studio 2010 with the .NET framework and with all the programming languages supported in Visual Studio.

Book II: Getting Started with Visual Studio

Use Book II to get up and running with Visual Studio 2010. If you already installed Visual Studio 2010 and are familiar with making your way around the Visual Studio interface, you can fast-forward through most of Book II. You find out how to install and navigate Visual Studio 2010 and use Visual Studio to say in touch with the Microsoft community.

Book III: Coding

Book III shows you all the major Visual Studio features for designing, writing, and generating code. Whether you're an experienced or novice programmer, you're likely to come across something you haven't seen yet. In Book III, you use the code editor to create source code and explore the basic language syntax of C# and Visual Basic. You also get a primer in object oriented programming and explore the class anatomy in .NET. Book III also shows you how to use Visual Studio to debug, analyze, and test your code.

Book IV: Basics of Building Applications with Visual Studio 2010

Visual Studio 2010 is all about creating applications. In Book IV, you dig into the kinds of applications you can create. In Book IV, you discover how to use Visual Studio to create C++ as well as .NET Windows and Web applications. You also discover new additions to the .NET framework for threads and parallel programming. Finally, you discover deploying your applications on the cloud with Windows Azure.

Book V: Getting Acquainted with Data Access

Nowadays, all applications require access to data. Book V surveys the vast array of data access features in Visual Studio 2010. Even a seasoned ADO. NET programmer should take a fresh look at the new data access code-generation features in Visual Studio because they can help you to explore data controls, create and manage connection strings, access data by using objects in a class library, model data from XML documents, and use Visual Studio for SQL Server 2008 projects.

Book VI: Going the Extra Mile

Visual Studio 2010 provides many features that take your productivity to new levels. At some point, all developers need to explore the topics covered in Book VI, such as configuring and managing the build process, keeping track of your code versions with source code control, and deploying Windows and Web application. In Book VI, you also discover creating applications for the Facebook social networking site.

Book VII: Extending the Family

Visual Studio 2010 is the development platform for many exciting new technologies being introduced by Microsoft. In Book VII, you explore how to find add-ons from Microsoft and other vendors to extend the features of Visual Studio 2010. You also explore Visual Studio 2010 and Team Foundation Server.

About the Companion Web Site

A companion Web site provides additional material. You can find it at `www.dummies.com/go/vs2010`. What you'll find is a section for each small book. Inside each section are resources links and projects with the source code.

Icons Used in This Book

In a book stuffed to the gills with icons, the editors have decided to use — you guessed it — more icons. Luckily, however, the book's icon set acts as visual signposts for specific stuff that you don't want to miss.

TIP

Tip icons point out advice that can save you time, trouble, and, quite possibly, cash.

TECHNICAL STUFF

These tidbits are completely optional, but if you're really into the technical side of things, you'll find loads of interesting info here.

WARNING!

Always read the information next to this icon! These icons highlight pitfalls to avoid as you deploy your applications or put the power of Visual Studio 2010 into action.

REMEMBER

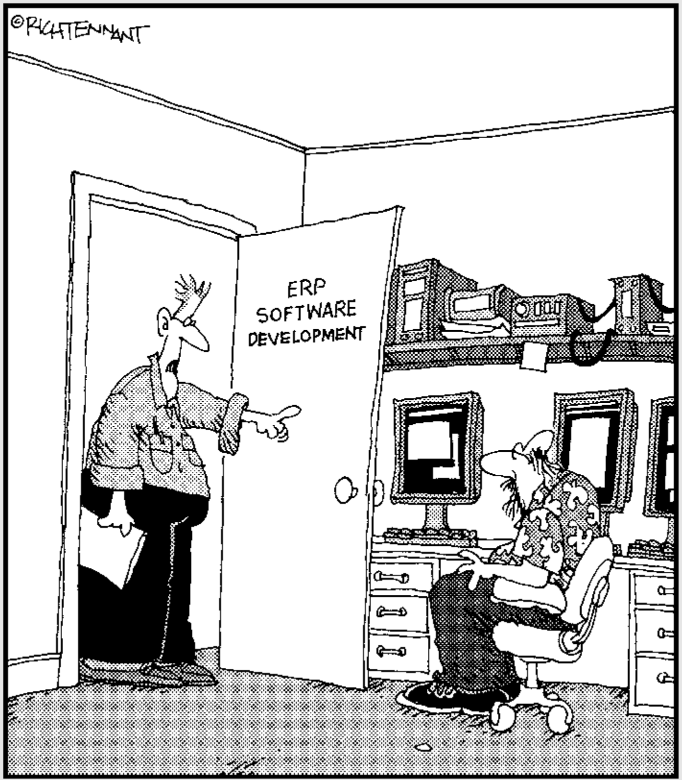
As its name suggests, this icon highlights stuff that you might want to, well, remember.

ON THE WEB

This icon points out resources that you can find on the book's companion Web site, which can help you further explore the topic being covered.

Book I

Visual Studio 2010 Overview



The 5th Wave By Rich Tennant

"Stop working on the Priority Parking Spot Allocation program. They want to fast track the Coffee Pot/Cubicle Proximity program."

Contents at a Glance

Chapter 1: What Is Visual Studio?

In This Chapter

- **Figuring out Visual Studio's role in software development**
- **Seeing Microsoft's vision for Visual Studio**
- **Saying hello to .NET**

To be truthful, building software that does more than just say something like "Hello world" requires more than just writing a few lines of code in a text editor. Who knew that business software could be so complex?

That's where tools such as Visual Studio enter the picture. Visual Studio enables you to build software more quickly by offering an advanced editor, compiler, and debugger in a single, easy-to-use package.

From Source Code to Application: How Software Is Written

There are three parts to writing software:

- **Creation of source code:** This source code is human-readable and normally text-based. Source code comes in many flavors depending on the language used. (Chapter 4 contains a lot more information about languages.)
- **Compilation:** During compilation, the source code is translated into binary executable data. This data takes many forms, including a compiler such as the one built into Visual Studio, an interpreter such as the command line (which ultimately creates binary executable data), or a variety of intermediate steps, such as a Java Virtual Machine, which takes pseudocode and converts to binary executable.
- **Execution of the program:** This step takes place as part of the development process while testing and then independently when users run the software. Figure 1-1 displays a summary of this process.

ON THE WEB

The *Visual Studio 2010 All-In-One Desk Reference For Dummies* companion Web site at `www.dummies.com/go/vs2010` has a list of the most popular and common tools that you can download from Microsoft. You can also find a link to a video presentation that shows the process of creating and executing programs.

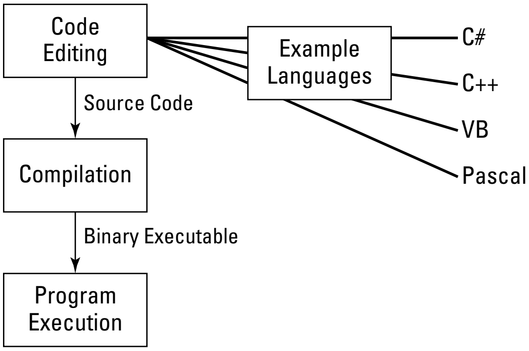


Figure 1-1: The process of creating and running programs is divided into three parts.

A *programming language* is a language that humans can understand and use to write computer programs. Computers understand only binary executable data, which is why a compiler has to translate the program from the human-readable programming language into binary executable. For .NET managed applications, the process is a little different as the compiler creates tokenized executable in the Microsoft Intermediate Language (MSIL), which the Common Language Runtime (CLR) then turns into machine code executable by the computer.

Technically, all you need to write source code is Notepad. Notepad would suffice for batch files, HTML applications, and even .NET programs. (The common joke at many seminars you attend is that Notepad is a developer's preferred HTML editor.) The .NET programs then can be compiled with one of the free .NET compilers, so why would you spend money on tools, such as Visual Studio, to develop software? The answer lies in the productivity that Visual Studio provides. It's one-stop shopping with a built-in editor, several compilers, and the ability to execute compiled programs. This type of program is known as an *Integrated Development Environment* (IDE). In addition, Visual Studio has a vast array of utilities, such as Spy++, and a world-class debugger that I talk about in Book III, Chapter 7.

You used to be able to access many programming tools from the command line. Using tools such as the Microsoft Assembler (MASM) (I still have a copy of it), I created many executables from the command line before IDEs even arrived on the scene. However, you can work with the .NET Framework without ever installing a copy of Visual Studio. (See the article at `www.devsource.com/article2/0,1895,1955461,00.asp` for a description of how you can create a .NET application using just Notepad and the .NET Framework. While this article covers JScript, you can use this technique for VB.NET and C#, too.)

Have you ever viewed an executable program (such as an .EXE or .DLL file) with Notepad? When you do, all you see is strange-looking characters. It's not human-readable because someone wrote the program with a human-readable language and then compiled it into machine code. You can get a better (or more accurate) view of machine code by using a hex editor like XVI32 (`www.chmaas.handshake.de/delphi/freeware/xvi32/xvi32.htm`) or a disassembler, such as Debug.exe (look in the \WINDOWS\system32 folder). However, the managed code that Visual Studio outputs is tokenized Intermediate Language (IL) code, not native code. Consequently, you use ILDASM.exe instead to view the output.

Just to be completely accurate, .NET language compilers don't exactly create machine code. Technically, they create intermediate code (or Microsoft intermediate language — MSIL). It's an intermediate step to machine code that is still not human-readable. When the operating system (OS) encounters MSIL code, it performs a compilation before the application actually runs. This process is known as *just-in-time (JIT) compilation.*

The Role of the Integrated Development Environment

Language compilers and all the tools you need to start writing computer programs are available for free. Most vendors provide software development kits (SDKs) that provide all the tools you need to edit code and compile it. But the Visual Studio IDE offers much more than an easy way to edit, compile, and execute programs. The following list shows reasons that Visual Studio should be carefully considered as a significant productivity boost to Notepad programming.

- An integrated debugger
- IntelliSense
- Project management
- Templates
- A comprehensive help system

One of the most popular SDKs for software development supports using the Microsoft family of languages, usually either C# or Visual Basic. You can download the Microsoft .NET SDK for free from the Microsoft .NET Framework Developer Center at `http://msdn.microsoft.com/en-us/netframework/aa569263.aspx`.

TIP

If you have Visual Studio 2010 installed, the SDK is already installed.

SDKs are an inexpensive way to play around with new tools or keep your skills updated in new or unfamiliar technologies.

Tools beyond the basic three

Developing software requires all kinds of tools. At the minimum, you need a text editor and a compiler, as described in the section “From Source Code to Application: How Software Is Written,” earlier in this chapter. As you might guess, however, other tools greatly assist development. In addition to an editor and compiler, you need tools for

- **Testing and debugging:** You can step through your code one line at a time to resolve errors with a debugger.
- **File linking:** Link all the files you need to build an entire application.
- **Configuration management:** Manage the configuration settings for everything from file locations to compiler settings.
- **Code editing:** Write code without having to memorize the syntax for a programming language. These tools, which have some intelligence about how to use the program language, can provide suggestions or context-sensitive help as you write code.
- **Deployment:** Easily deploy your compiled application to other computers, where it can be executed.

When you download an SDK, you get some of the preceding tools, as well as many others. The problem you quickly discover is that managing all these individual tools is cumbersome. Plus, many of the tools are command-line tools — you get no point-and-click convenience here.

That’s when the idea hits you: “Hey, wouldn’t it be great if all these other tools were integrated into a single tool?” That’s exactly what an IDE does: It puts all the individual tools — the intelligent code editor, the debugger, and the configuration manager — into a single tool where they can interact with one another.

Enter Visual Studio

The Microsoft IDE is Visual Studio. Visual Studio 2010 is the latest version of this product. Some editions of Visual Studio are better suited to individual developers, whereas others are geared toward developing software in a team setting. (See Book I, Chapter 5 for more details on these editions and how to choose the edition that’s right for you.)

With Visual Studio 2010, you get these features:

- **Programming languages, such as Visual Basic .NET and C#:** See Book I, Chapter 4 for more details on the languages you can use with Visual Studio 2010.

- **Technologies for building high-quality software, such as Windows applications, Web-based applications, Web services, and applications for mobile devices (such as smartphones):** Book III shows you how to use Visual Studio 2010 to build applications.
- **Data access tools that allow you to access data from any of the popular database management systems, such as Microsoft SQL Server or Oracle:** You can also access text files and XML files. Book IV covers the data access capabilities of Visual Studio 2010.
- **Tools for debugging, designing, testing, and deploying applications:** Book VI covers many of these tools.
- **All the features of the Microsoft .NET Framework, which provides a rich set of features that allows you to work at a higher level of abstraction:** Book I, Chapter 2 discusses the evolution of .NET, and Chapter 3 describes the services of .NET.

You aren't restricted to using Visual Studio just for Microsoft .NET, nor is the popular open source editor Eclipse limited to Java. You can use Visual Studio to create Java applications, or you can create .NET applications by using Eclipse.

You can use multiple programming languages, thanks to the nature of IDEs. All the various tools that are integrated into an IDE can be created by multiple vendors. Instead of restricting you to just the set of tools that comes with a particular IDE, a *plug-in* allows you to use additional tools created by third-party vendors or the open source community.

To download the Visual Studio plug-in named Grasshopper (which allows you to create Java applications), go to `http://dev.mainsoft.com`. You can find plug-ins for Eclipse at `www.improve-technologies.com`.

Why would you want to mix up your languages with your IDE? The answer is productivity. In the same way that using a single tool (such as an IDE) is better than using 12 different tools, using an IDE that you're already familiar with is more productive than switching to a new IDE. So, if you work in a company that develops Microsoft applications primarily using Visual Studio, you can also use it to create Java applications. You don't need to learn to use a whole new tool, such as Eclipse.

ON THE WEB

A podcast that talks about the compilation process is available on this book's companion Web site. You can find the link from the page at `www.dummies.com/go/vs2010`.

Visual Studio as the Hub

As if the integration of tools weren't enough, Microsoft has something else in mind for Visual Studio: It envisions Visual Studio as the hub for all the server applications with which a developer might interact.

For example, instead of accessing Microsoft SQL Server by using the tools for that server, you can access SQL Server features right inside Visual Studio. In the future, Visual Studio will integrate with even more server applications. In this way, Visual Studio is the hub of all your interactions with your company's information technology environment — assuming, of course, that you're strictly a Microsoft shop.

Microsoft has another kind of hub in mind for Visual Studio. The company realized that software development involves more than just coding like a madman. In fact, writing code usually involves only one-quarter to one-third of the effort involved in building software. The rest of the project's time is spent gathering and analyzing requirements, creating models that explain those requirements, and refining those models into tangible designs that are ultimately translated into code. After the code is written, the software has to be thoroughly tested and bugs tracked and maintained.

Many developers use everything from pen and paper to third-party tools to perform the rest of the tasks involved in building software. Microsoft saw an opportunity to bring all this activity under the Visual Studio umbrella. In the past, a developer may have used Visio to create models and NUnit to automate code testing. Now, all these tools are integrated in Visual Studio, and they work together.

The tools work together so well that a model you create in Visual Studio can even generate code. Visio could do that, for example. The model in Visual Studio, however, updates to reflect changes that you make in the code itself!

Microsoft also realized that software development doesn't happen in a vacuum. The days of hotshot developers isolating themselves for weeks at a time to build the next big applications are long gone. Companies have finally realized that this approach to software development isn't sustainable.

Instead, a company usually has a team of developers, often with specialized roles, working on a project at the same time. By acknowledging that software developer means more than just programmer, Microsoft has expanded Visual Studio with its Team Foundation version to become the hub of a company's team development activities. (See Book I, Chapter 5 for more information about the different versions of Visual Studio.)

A team using Visual Studio can log a software bug, associate the bug with a section of code, assign the task of fixing the bug to a developer on the team, and track the resolution of the bug. All this happens in Visual Studio!

Microsoft's goal (in addition to increased profits) is for Visual Studio to become the personal productivity tool for software developers. In the same way that Microsoft Office has increased the productivity of office workers by freeing them from typewriters and calculators, Microsoft intends for Visual Studio to create a new standard of productivity for individual software developers and the teams on which they work.

The Keeper of .NET

The .NET Framework is the Microsoft platform for creating modern Windows, Web, and mobile software applications. Although the platform provides the steam that makes your applications go, the Visual Studio 2010 development environment allows you to harness that power. The power of .NET is made accessible by Visual Studio, and its widespread use wouldn't be possible otherwise.

Back in the good old days of software development, before frameworks like .NET existed, developers had to write a lot of code to do simple tasks, such as open a file and display its contents on the screen. In an effort to simplify repetitive tasks, many programming languages started providing helper functions that developers could call. Development environments, such as Visual Studio, were often tailored for use with a specific programming language, such as Visual Basic.

Helper functions were created to do just what their name implies: Help developers do something they needed to do. Instead of having to interact with the OS and telling it to find a file and open it for reading, all a developer had to do was call a helper function and then tell the function the name and location of the file to open. The helper function would then "talk to" the OS and return the file's contents. The developer could use another helper function to display the returned contents on-screen. If the developer then decided to give the user the option to print the file's content, another helper function handled all the details of printing the file.

These helper functions improved a developer's productivity and enabled that person to work at a higher level of abstraction. Over the years, companies such as Sun and Microsoft that make software development tools realized that developers needed a little more than just helper functions. Common software development problems had to be solved, such as how to

- Manage memory
- Ensure that code is secure
- Allow programs to be moved easily from one hardware platform to another, such as from Windows to Linux

The solution to this problem was to create a virtual hosting environment in which software applications could run. This host, also known as a virtual machine or runtime engine, provides services such as file IO and memory management to the software that is executed inside the host. The Sun version of the virtual machine is Java. The Microsoft version is the .NET Framework, also referred to as just .NET or Microsoft .NET.

The .NET Framework is more than just a simple set of helper functions. Applications that are created with the .NET Framework are hosted in a virtual machine called the CLR. Therefore, before a computer can run an application that you build by using .NET, the .NET Framework must be installed on the computer. The version of the framework that's installed is the .NET Framework Redistributable, which is a free download from the Microsoft Web site. Many new computers have the .NET Framework Redistributable already installed, and most corporations are installing the Redistributable on all their computers.

By running your application in the .NET framework, your application can take advantage of the many services that .NET provides.

You can download the Microsoft .NET SDK to get the tools you need to start building software. To fully capitalize on all the development features of .NET, however, you need Visual Studio.

Book I, Chapter 6 shows you the new features of Visual Studio 2010. They're not just new features, though; some can be seen as new paradigms, such as the Cloud Computing support.

Chapter 2: Exploring .NET

In This Chapter

- **Discovering how .NET has evolved**
- **Sneaking a peek at the components of .NET**
- **Looking into the future of .NET**

In an attempt to mitigate the increasing complexity of building software, Microsoft released a new set of tools in 2002 (although the first beta was released in July 2000) for building software: the Microsoft .NET Framework.

The .NET Framework provided the plumbing necessary to write complex applications. In the past, I remember getting some really tough assignments with the expectation that they would be done in a few weeks. Although the business logic was fairly easy, the infrastructure was not. Some of the more complicated assignments I encountered included programs that had to communicate over the Internet, create graphically appealing reports, or perform complex database access. At that time, those tasks were difficult. Now, the .NET Framework makes them easy — almost trivial.

.NET is a reflection of the latest thinking about, and best practices for, how software should be developed. Visual Studio is the premiere toolset that Microsoft created for developing software by using the .NET Framework. Although the road to Microsoft .NET and Visual Studio has been misunderstood by some at times, most developers agree that using Visual Studio to develop Microsoft .NET applications is a huge productivity boon.

Following the Evolution of .NET

Microsoft released the first version of .NET in 2002. Because Microsoft tried to append the .NET moniker to all its initiatives, from software development tools to enterprise servers to operating systems, .NET initially suffered from an identity crisis. Thanks to the ubiquitous use of the term by Microsoft, however, .NET is now as much a brand as it is a technology.

Six versions of .NET have been released. Each of these versions represents a milestone in developing software with .NET:

- **Version 1.0:** Released in 2002 with the Visual Studio .NET integrated development environment. In version 1.0, all development — whether

Windows-based or Web-based, and regardless of the language — was integrated into Visual Studio. Prior to the release of .NET, each Microsoft development tool was a separate product.

The object-oriented language C# was created as part of .NET. Visual Basic was completely revamped to be object oriented. Many Visual Basic developers felt betrayed, and Microsoft has had a hard time convincing them to make the leap.

Data access was greatly simplified with ADO.NET, and ASP.NET was introduced for developing Web applications. Even though these technologies share the same names as their predecessors (ADO and ASP), the resemblance stops there. Like all .NET technologies, ADO.NET and ASP. NET are object oriented.

- **Version 1.1:** Released in 2003 with Visual Studio .NET 2003. With version 1.1 of .NET, many 1.0 features that either weren't ready yet (such as the Compact Framework for devices such as the PocketPC) or were available separately from Visual Studio (such as ASP.NET) were completely integrated into Visual Studio .NET 2003. Version 1.1 was more stable and more widely accepted. During this time, the Microsoft .NET brand became diluted from overuse.
- **Version 2.0:** Released in October 2005 with Visual Studio 2005. In the two years between the release of versions 1.1 and 2.0, the *dot-net community* — as the collective of .NET developers is often called — busily wrote applications that demonstrated how powerful .NET could be. Microsoft listened, and many of the suggestions for ways to extend .NET, which were written about in articles on Web sites such as `www.gotdotnet.com` and `www.4guysfromrolla.com`, were implemented as new features in the 2.0 version of .NET. As of this writing the GotDotNet site has been shut down, and the MSDN Code Gallery at `http://code.msdn.microsoft.com` has replaced it.

  For the release of .NET 2.0, Microsoft dropped the .NET suffix from its nondevelopment software. .NET now refers to just the application platform — the .NET Framework.

  Visual Studio 2005 matured into more than a mere coding environment and can now manage many aspects of the software development lifecycle. The Team System version of the product uses a server-based component as a data repository. C# established itself as a rock-solid, object-oriented language. MSDN, the Microsoft Developers Network, became more tightly integrated with the Visual Studio product.
- **Version 3.0:** Released in November 2006 with the Windows Vista OS. Version 3.0 combined version 2.0 with four new technologies:
  - **Windows Presentation Foundation (WPF):** Gave developers the capability to create much richer applications by providing better controls. It helped separate the user interface (UI) from the business logic. It also assisted developers who wanted a consistent UI between desktop applications and Web-based applications.

- **Windows Communication Foundation (WCF):** Provided easier and more robust communications between connected systems. It is built on the Web services technology.
- **Windows Workflow Foundation (WF):** Enabled developers to create workflows by using a new XML-based language named XAML.
- **Windows CardSpace:** This security framework helped users to manage identities that can be used in transactions such as Web site logins.

- **Version 3.5:** Released in October 2007 with Visual Studio 2008. Version 3.5 builds incrementally on version 3.0. In addition, it adds the following:
  - Language Integrated Query (LINQ)
  - Integrated use of Asynchronous JavaScript and XML (AJAX)
  - Full support for WF, WCF, and WPF
- **Version 4.0:** Released in April 2010 with Visual Studio 2010. Version 4.0 builds on version 3.5 and adds the following:
  - Parallel processing
  - Entity Framework
  - Support for cloud computing
  - The F# programming language

With the Microsoft .NET, Microsoft developers can stand toe-to-toe with Java developers and know that they can use almost all the same language and development features. And, with Visual Studio 2010, many developers' favorite third-party tools are part of the IDE, which boosts productivity immensely. At the same time, the Microsoft Team Foundation edition of Visual Studio 2010 draws a line in the sand, positioning Visual Studio and Microsoft .NET as the tools of choice for enterprise software development.

Getting Acquainted with the .NET Components

To fully grasp how all the pieces of .NET work together, you first need a basic understanding of all those pieces. At a very high level, you can think of .NET in terms of

- **The .NET Framework:** All the software that makes Visual Studio possible
- **The .NET Software Development Kit (SDK):** All the software that enables you to build software applications by using the .NET Framework

TIP

You can download both the .NET Framework and the SDK for free from Microsoft at `www.microsoft.com/downloads/details.aspx?FamilyId=AB99342F-5D1A-413D-8319-81DA479AB0D7&displaylang=en`.

- **The programming languages of .NET:** The four languages that Microsoft offers (in addition to dozens more that are available from third parties):
  - VB.NET (Visual Basic)
  - C#
  - C++
  - F#

See Book I, Chapter 4 for more on these languages.

A language that can be used to develop .NET applications is said to target the .NET Framework.

- **The technologies of .NET:** All the stuff you can build with .NET, such as Windows and Web applications, Web services, and mobile devices, such as smartphones
- **Visual Studio:** The IDE that puts access to everything in this list at a developer's fingertips

The default languages of .NET are also free in the sense that you can learn the language syntax and use any text editor to write it. The compiler is the tool that you need to convert the programming language to a language the machine can understand, and it's also free. The compiler is part of the .NET Framework. (If you install a third-party language product, then that third-party product will have its own compiler and may not be free.)

To take full advantage of the languages and technologies of .NET in a seamless way, however, you need Visual Studio.

.NET freebies

To start developing .NET applications, all you need is the .NET Framework and the .NET SDK. The .NET Framework provides the bulk of the functionality; however, you need the SDK in order to write code for the .NET Framework.

The Microsoft .NET Framework consists of these three free components, which become resident when the .NET Framework is installed:

- **Common Language Runtime (CLR):** The CLR provides a managed environment for running software applications. Developers access the features of the CLR through the Base Class Library (see later bullet). Some features of the CLR, such as automatic memory management, just happen. (The developer doesn't do anything.)
- **Common Type System (CTS):** The CTS sets the rules for how programming languages that use the .NET Framework must behave. New languages can then be developed for use with .NET.

- **Base Class Library (BCL):** The BCL is the part of the .NET Framework that developers get to "touch." The BCL enables developers to create Windows- and Web-based UIs and to access databases and services of the operating systems.

Because the .NET Framework is complex, a detailed discussion of every aspect of the Framework is beyond the scope of this book. Knowing all the details of the inner workings of the .NET Framework isn't necessary for most developers, anyway, which is part of the reason why Visual Studio is such a valuable tool. It allows developers to access the features of the .NET Framework at a higher level of abstraction and "hides" the implementation details.

The .NET Framework SDK provides developers with free tools they can use to access the .NET Framework. In addition to some general utilities, the SDK contains tools to help you

- Manage configuration and deployment and configure the .NET Framework
- Debug software and manage security

Language compilers are installed with the .NET Framework.

If you're using Visual Studio 2010, the SDK is installed by default. You might still want to use some of the utilities in the SDK to automate build processes or perform other functions that aren't part of the Visual Studio interface.

If you really want to learn how the .NET Framework works, dig into the SDK and use the Framework command line compilers. In other words, do some development without using Visual Studio. You can gain a much better appreciation of all the time that Visual Studio saves, and you can better troubleshoot problems that occur in Visual Studio.

No free lunch

Yes, the fundamental components of .NET are free. However, if you expect to be productive, you need to spend some dough. Specifically, get your hands on Visual Studio 2010. With Visual Studio, you're better positioned to take advantage of the languages and technologies of .NET. You can build some significant applications using the Express Edition of the product.

Dozens of languages for .NET are available, although only these four are from Microsoft:

- **Visual Basic:** Visual Basic has a streamlined syntax that many developers find easy to learn and use.

- **C#:** C# is a rapid application development language. Some of the .NET Framework is written in C#.
- **C++:** You can use this more advanced language to build business applications, operating systems, and language compilers.
- **F#:** F# is a multiparadigm programming language for the .NET Framework

Here are a couple of third-party languages available for .NET:

- Borland Delphi
- COBOL

For a fairly complete list of all the languages available for .NET, go to `http://en.wikipedia.org/wiki/Microsoft_.NET_Languages`.

For more specifics on the languages of .NET and how to choose and learn one, see Book I, Chapter 4.

The technologies of .NET are what allow you to build cool applications. With these technologies, you can

- Build Windows applications by using Windows Forms.
- Build Web-based applications by using ASP.NET.
- Access data from all different kinds of data stores by using ADO.NET.
- Share data between disparate systems by using Web services.
- Create software for mobile devices, such as smartphones, by using the .NET Compact Framework.

Peering into the Future of .NET

The world of application development has changed significantly in the past decade. The concept of hosting software inside a runtime engine is now the de facto standard for building most software applications — and Microsoft .NET is a major player.

Visual Studio is all grown up from its days as a meager code editor. You can use it now to manage many aspects of software development. As the keeper of .NET, Visual Studio is poised to take .NET into more aspects of the software development life cycle, such as designing and testing.

As Microsoft looks to the future, it's placing its bets on its new OS, Windows 7. Windows 7 improves Windows Vista, which is completely different from any of the previous Microsoft Operating Systems, features these completely redesigned subsystems:

- A new presentation layer for building Windows applications, graphics, videos, and documents, called Windows Presentation Foundation (WPF): Applications that target the WPF are called Aero applications, the name of the new Vista user interface. Aero might eventually replace Windows Forms, but that replacement is several years away.
- A new communications subsystem called Windows Communication Foundation (WCF) that manages all communications and frees developers from having to handle the details. The subsystem also supports peer-to-peer communications. You can still use good old-fashioned Web services if you want, but all WCF communications are integrated with Vista.
- A programming model using the .NET Framework to easily access all features of Windows: Previous versions of Windows used a C library interface that was challenging to learn. Previous versions of the .NET Framework provided only limited access to the Windows OS. Microsoft is positioning .NET as the programming model of choice for Vista and Windows 7 applications, which makes it possible to
    - Deploy applications to Windows computers and better manage updates to those applications
    - Provide more secure applications by using the security features of Windows
    - Develop more stable applications

Another exciting area of development is Microsoft Office 2007. Microsoft has expanded the Office feature set to include new server products, such as SharePoint Portal Server. The new name for SharePoint Portal Server is Microsoft Office SharePoint Server 2007. Office includes servers for InfoPath forms and Excel reporting. Of course, all these features are extensible with Microsoft .NET.

Microsoft even released a special edition of Visual Studio: Visual Studio Tools for Office (VSTO). VSTO allows developers to create applications by using the .NET Framework and Office.

The next versions of Visual Studio are expected to more tightly integrate Visual Studio and Microsoft .NET with the new features of the OS. You'll still be able to do all the traditional business development by using Windows Forms.

Chapter 3: Modern Software Development

In This Chapter

- **Understanding the professional approach to software development**
- **Building a software development toolbox**
- **Seeing why component-based software is superior**
- **Peeking at how programs are executed in .NET**
- **Discovering automatic memory management**

This chapter offers an abbreviated history of software development by discussing where the process now stands and how it got there. You also find out about some of the inner workings of .NET as it relates to modern development approaches.

Examining the Software Development Process

Even if you've never worked in software development, you've no doubt heard about many of the problems associated with software development projects, such as

- Frequent schedule and budget overruns
- Software that doesn't do what it's supposed to do
- Software that's obsolete by the time it gets finished
- Software that ships with security flaws

Over the years, the software development industry has created a number of approaches and tools to help better manage software development. I discuss several approaches in more detail in the upcoming section "Have process, will repeat." However, some of these approaches can be categorized in one of these two main approaches, although most real-life situations are a hybrid of the two:

- **Ad hoc, or hotshot:** In this approach, characterized by nonexistent development practices, no standard exists for how software is developed. Hotshot programmers often work all night and come in at noon the next day. You usually get no documentation, and only the programmer understands how the code works.

- **Rigorous and slow:** As a rejection of the other approach, this one takes an engineering view of software development. The argument is that dotting all your *i's* and crossing all your *t's,* and then getting proper sign-off at each step along the way, gives software development a process that's repeatable and — most importantly — accountable. Unfortunately, the engineering approach adeptly creates mountains of documentation but doesn't adeptly create working software on budget and on time. It also squelches the creativity that truly phenomenal innovations rely on.

After producing many failed projects, software developers have started to get the message that ad hoc approaches are too immature and the rigors of an engineering mentality probably too inflexible. At the same time, most companies have learned that relying on hotshot developers is a recipe for disaster. Instead, many firms now hire developers who write manageable code, rather than someone who can write code faster than anyone else.

Of course, in spite of everyone's best efforts, sometimes a superhuman effort may be required for several days. One particular project that comes to mind involved training that was to take place on Monday. On Saturday, several severe software flaws were discovered that had to be fixed. They were all fixed by Monday at 2 a.m. But, as you can likely guess, no one got a lot of sleep that weekend.

Looking at Software Engineering in Action

Over the years, a number of methodologies and tools were created in an attempt to solve the problems involved in software development, with each solution promising to deliver software development projects from the evils of scope creep and second system syndrome. (*Scope creep* occurs when additional features that weren't originally planned for find their way into the software.)

If you've ever worked on the second incarnation of an existing system, you've no doubt experienced *second system syndrome.* In this case, the project quickly gets bloated because developers and users see so many opportunities to improve the existing system. Unfortunately, many methodologies and tools designed to keep projects on track are not only very expensive, but their implementation also requires significant overhead. As a result, companies spend a lot of money but don't always see results. And, on a personal note, any time that the marketing department has input, runaway feature creep is a real danger.

Now that software developers have had some time to see what doesn't work, they've started to create a living framework of what does work:

- **Repeatable processes:** Rather than treat every project as if it's shiny and new, developers have started to acknowledge that they should take some common steps. Also, by acknowledging that different kinds of software development projects exist, developers are rejecting the notion of a one-size-fits-all process and are instead creating processes that can scale up or down to fit the project.

- **Best practices:** By creating a body of industry best practices, software developers are sharing and reusing their knowledge of how to solve problems.
- **Multiple tools:** Developers have learned that no single tool can do the job. Instead, tools from many different vendors and the open source community is a better approach than putting all your eggs in one basket with a single vendor.

You can find a list of best practices at this book's companion Web site. You can link to it from the page at `www.dummies.com/go/vs2010`.

Have process, will repeat

When they're ready to start a new software development project, many developers aren't sure where to begin. Depending on your role in the project, the project may not actually start for you until after somebody else completes his responsibilities. Does the project start before or after the kick-off meeting? Should you even have a kick-off meeting — and, if so, who should attend? These questions are answered by the management style and processes that your company adopts for developing software.

Managing a software development project tends to fall somewhere between these two approaches:

- **Waterfall:** In the waterfall approach, one phase of the project completely ends before another phase can start. In this way, the project "waterfalls" through the stages.
- **Iterative:** Projects that are developed iteratively go through the same phases multiple times. Developers can move backward and forward through the development phases several times to iteratively develop their understanding of both the problem and the solution.

A process for developing software is like any other process: It's a set of steps you go through to achieve an outcome. Your company has a process for paying bills and entering into contracts. Some business processes are more strictly defined and closely adhered to than others. The same is true for developing software. Some companies have strict rules about how projects are initiated and how they progress. Other companies approach each project like it's their first.

This list offers a few of the popular process approaches:

- **Software engineering:** The rigorous engineering approach brings an engineering mindset to software projects. A body of standards outlining the kinds of documentation and processes that should support a project is defined in the Institute of Electrical and Electronics Engineers (IEEE) software engineering standards.
- **Agile:** This approach embraces change as part of the software development process. In most approaches, change is usually considered a bad word. Agile developers work in pairs, create many prototypes of their solutions, and incorporate user feedback throughout the entire process.
- **Test-driven:** This newer approach to building software involves building test harnesses for all code. These tests, written from the requirements documentation, ensure that code can deliver promised features.
- **Rational Unified Process (RUP):** The RUP commercial process uses the IBM suite of Rational tools to support software development. Its noncommercial equivalent is the Unified Process (UP). RUP and UP are iterative approaches.

What's common to all these approaches is the acknowledgment of a software development life cycle (SDLC). Most processes account for the fact that all software progresses through a life cycle that has similar stages:

- **Initiation:** Project planning and justification get the project up and running.
- **Requirements gathering:** After the project is approved, developers start talking with users about what they expect the software to do. Frequently, this phase requires re-examining the planning and feasibility efforts from the preceding stage.
- **Analysis:** At this stage, developers analyze the requirements they have gathered to make sure that they understand what end users want.
- **Design:** Developers start to work out models that describe what the requirements might look like in software.
- **Construction and implementation:** Developers write the programs that execute the models created in the design stage.
- **Testing:** The programs are tested to ensure that they work properly and meet the requirements that the end user specified.
- **Deployment:** Software is installed, and end users are trained.
- **Maintenance:** After the software has passed all its tests and been implemented, it must still be supported to ensure that it can provide many years of service to the end user community.

In a project that uses a waterfall management approach, the software is likely to progress through the stages of the life cycle in chronological order. A project managed iteratively cycles back through the phases while it makes forward progress. On larger projects, people usually fill specialized roles at different stages of the SDLC. One person often fills multiple roles. Table 3-1 lists some common roles in the SDLC.

Table 3-1 Roles in a Software Project

SDLC Phase	*Role or Job Title*
Initiation	Project manager, project sponsor
Requirements gathering	Analyst, subject matter expert
Analysis	Analyst, subject matter expert
Design	Designer, architect (possibly specialized, such as software architect or technical architect)
Construction and implementation	Programmer
Testing	Tester, quality assurance (QA) personnel
Deployment	Installer, trainer, technician, technical architect
Maintenance	Support personnel, programmer, tester

A software project generally doesn’t flow like a waterfall through stages of the SDLC, so that’s a big reason why the waterfall approach has fallen out of favor. In most cases, the stages of the SDLC are either

- **Iterative:** The stages are repeated multiple times throughout the project. For example, developers commonly loop through the initiation, requirements, and analysis phases several times until they get a handle on what they’re expected to build. Each loop delves progressively deeper into the details. You’ll frequently question the assumptions of the requirements and analysis stages after you start building models in the design stage.
- **Concurrent:** Stages often occur at the same time. For example, a group of developers might analyze requirements while another group starts to build the design models. The design group can start to validate (or invalidate) the analysis assumptions.

TIP

Choosing a process isn’t an either/or proposition. Many teams like to combine the features they like best from several different approaches. For example, your team might like the change-management features of the agile approach. If you have doubts as to which way to go, you can refer to the IEEE software engineering standards.

Regardless of the approach you decide to take, here's some advice to take into consideration:

- **Use a process — any process.** Homegrown processes are sometimes better than store-bought ones. If everyone gets to participate in creating a process, they're more likely to want to see it succeed.
- **Repeat the process.** A process becomes a process only after it has been repeated multiple times. Doing something once and then abandoning it doesn't create a process.
- **Improve the process.** You can't improve your process unless it has some means of providing feedback and metrics for measuring. Without metrics, you have no way of knowing when something has improved. If you're unsure how to approach collecting project metrics or improving your process, take a peek at other methodologies to get some ideas.

Practicing what you preach

A body of knowledge has started to develop around software development. Developers realize that many businesses are trying to solve the same kinds of problems. Rather than treat each project as a one-off experience, developers know that elements of the project are common to other projects that they need to develop. When a problem gets solved the same way over and over again, the solution is often referred to as a *best practice.* For a best practice to develop, many people must be trying to solve the same problem. For example, the following list includes several problems that Web site developers try to solve:

- Providing secure access
- Logging errors
- Accessing data

Businesses also need to solve a common set of problems, such as the following:

- Capture and fulfill orders
- Manage customer information
- Invoice and receive payments

Rather than try to solve all these problems on your own, an entire community of developers, authors, and companies are sharing their experiences and guidance on how to approach these problems.

This shared knowledge can be called best practices, design patterns, frameworks, or models. Microsoft's own vision for sharing knowledge includes patterns, practices, and software factories.

In fact, the underlying premise of runtime environments, such as the Microsoft .NET Framework and the Java Virtual Machine, is that all programs need common features, such as portability and memory management. Frameworks, patterns, practices, and models are all knowledge repositories that enable developers to reuse the work of other developers. Modern software development makes extensive use of all these knowledge-sharing tools.

According to Professor Joe Hummel, of Lake Forest College, some of the best practices for software development include these elements:

- **Object-oriented programming:** This style of programming makes programs easier to understand and test.
- **Components:** Software that's broken into components is easier to deploy. The upcoming section "Components Defeat Monoliths" expands on the benefits of using components.
- **Testing:** *Integration testing,* or repeated automated testing, is important in order to see how components work together.
- **Code reviews:** Standards are crucial for identifying how code should look. Having peers review each other's code enforces standards and exposes developers to new styles. (As the adage goes, "Two heads are better than one.")
- **Prototyping:** You should build your software frequently, not only to make sure that it works, but also to get it in front of end users early and often so that they can validate your progress.
- **Tools, tools, tools:** Use tools to help you manage your process and projects. See the next section for more about using tools.

Another best practice to keep in mind is that when it comes to developing software, less is more. A team of four to six developers can usually develop higher-quality software than larger teams can. As team size increases, so does the complexity of keeping everyone in the loop. Keep your team sizes small.

When you're using best practices, you have to keep things in context. The tools and processes you use for building commercial-quality software may not be the same tools and processes you should use to build software for an internal department that has a closed information technology environment.

Building a developer's toolbox

Just like plumbers and auto mechanics have toolboxes, software developers have toolboxes, too. Your software development toolbox should include all the usual suspects:

- Integrated development environment (IDE), such as Visual Studio

- Third-party tools that you like to use for testing or logging

  TIP

  Many developers take advantage of open source software, such as NUnit for unit testing and Log4Net for logging application errors.

- Project management software, such as Microsoft Project, or maybe even a simple spreadsheet program
- Collaboration software, such as Windows SharePoint Services, that you can use to store your project's artifacts
- Source code-control software so that you can keep all your source code safe and secure. Some popular source code control applications are Perforce, Microsoft Team Foundation Server, and Subversion.

You should build a toolbox of resources to which you can turn for advice about how to handle security, build secure Web sites, and face any other kind of software challenge. Sure, you could just Google whatever you're looking for. You have more success, though, if you turn to a common set of resources. Here's a start:

TIP

You may have more success using a Google site search for something on the MSDN or the Microsoft Knowledge Base by going to www.google.com/advanced_search?hl=en and providing an entry in the Search Within a Site or Domain Field.

- **Software Engineering Institute (SEI):** The SEI plays a huge role in defining software engineering. You can find SEI on the Web at www.sei.cmu.edu.
- **Software Engineering Body of Knowledge (SWEBOK):** The SWEBOK represents the consensus among academicians and practitioners for what the processes for software development should look like. If you have never thought about what it means to engineer software, the SWEBOK is a great place to start. You can download it at www.swebok.org.
- **Microsoft patterns and practices:** A few years ago, Microsoft finally started sharing with the world how it thinks that software developed by using Microsoft tools should be developed. Microsoft patterns and practices is a combination of books, articles, software, and other resources that help you write software the way Microsoft believes it should be written. Find patterns and practices at http://msdn.microsoft.com/practices/GettingStarted.
- **Rational Unified Process (RUP):** RUP is one of the more popular development processes. You can find tons of books and articles on the subject. Note that you can use the RUP without buying any of the tools that IBM makes to support the process.
- **Agile Manifesto:** The manifesto for agile software development can change the way you think about software development. Read the manifesto at http://agilemanifesto.org.

- **MSDN Webcasts:** The Microsoft Developer Network, or MSDN, presents Webcasts on every kind of software subject imaginable. You can find Webcasts at `http://msdn.microsoft.com/events`. Read more about MSDN in Book II, Chapter 3.

  A Webcast that you may find valuable is the series Modern Software Development in .NET, by Professor Joe Hummel. Dr. Hummel has a series on C# and on Visual Basic; you can download his Webcasts from `www.microsoft.com/events/series/modernsoftdev.mspx`.

Much of the advice you find tells you how software should be developed, but not so much how it is developed. Become familiar with how things are supposed to be done as you continue learning the techniques that other developers use to solve real-world problems.

Working with your partners in development

The most important lesson that software developers have learned is that “it takes a village.” Long gone are the days when hotshot programmers were expected, or allowed, to hole up in their offices for weeks while they created the next big application.

Now, software development is recognized as a team effort. Although some teams still place people in specialized roles, such as programmers, people are more commonly filling multiple roles on a project. The lines between analysts, designers, programmers, and testers are becoming more blurred.

The blurred roles are part of the reason that many developers objected to Microsoft’s planned release of different versions of Visual Studio based on the roles of a development team. As roles blur, developers are expected to be more flexible.

Not all developers work a 40-hour week, of course, although the length of that workweek is becoming the norm. Many consulting houses are becoming more reasonable about creating a work-life balance for their employees, and an increasing number of developers are maintaining reasonable hours. There are even rumors that developers at Microsoft, known for working extremely long hours, are now working regular hours.

Developers are sometimes expected to work extra hours, of course. Sometimes, during the software development life cycle, extra work is required, although it’s becoming more of an exception than the norm.

Components Defeat Monoliths

Older software that was developed before the creation of modern software development techniques, or that uses outdated practices or obsolete development tools, is sometimes called *legacy software.* Almost all businesses support at least a few legacy applications. Sometimes, a company's legacy software is limited to a few non-essential utilities used by a single department. At other times, the legacy application is a mission-critical application that must be kept running. Either way, at some point, you'll be asked to support a legacy application.

Here are some of the challenges you might encounter in supporting legacy applications:

- **Minimal, missing, or out-of-date documentation:** Although documentation usually isn't the strong suit of most developers, the complexity of some legacy applications can be overwhelming. Newer development techniques favor simplicity and allow the code to be self-documenting. At the same time, more intelligent tools, such as Visual Studio 2010, can update modeling documentation to reflect changes made in the source code.
- **Components that are tightly coupled:** A legacy application sometimes has dependencies on other software that can cause the application to crash. Components are considered tightly coupled when you can't separate them from each other. If the software is broken into pieces, those pieces, or components, can't be reused for other software. New applications must be created, which leads to the same logic being deployed in multiple applications.
- **The use of nonstandard protocols:** Software that was developed before the creation of standards, such as XML, often use proprietary file formats and communications protocols. This situation makes it difficult for the software to interoperate with other software.
- **Spaghetti source code:** The emphasis in coding hasn't always been on readability. As a result, programmers write programs that are cryptic and hard to understand. This can happen for two reasons. One, as you may expect, is from poor planning and a low skill level. The second, though, is because developers sometimes have to write code with limitations that are imposed upon them by situations, such as hardware limitations.
- **General deployment and maintenance difficulties:** Many legacy applications were intended to be deployed in much simpler hardware configurations than the ones in use now. Software written in the mid- to late 1990s was probably never intended to be used over the Internet, for example.

You can, of course, find software developed by using all the latest tools and practices that have all these characteristics. However, developers who are paying attention to industry best practices and working with, rather than against, modern tools should find that they have to go out of their way to write bad software.

Software written with these drawbacks is monolithic. With a monolithic application, you can imagine the developer sitting down and writing one long piece of source code that does 25 different things. Although the software might work and be efficient, this approach is hard to maintain and scale.

Software developers now focus on

- **Readability:** Writing software shouldn't be a contest to see how cryptic you can be. Today's approach favors a programming style that avoids shortcuts and abbreviations and instead uses an implementation designed to be transparent to future developers. This technique makes the code self-documenting because you don't have to write additional documentation that explains what the code does. One thing to note is that the compiler is very smart when it comes to optimizing the output. Most of the time, clearly understandable code compiles to the same thing that seemingly efficient but cryptic code compiles to.
- **Components:** Breaking down software into discrete, manageable components is favored over creating one huge application. Components can be reused in some cases and are easier to deploy.

One reason why these changes have come about is that management has realized that hiring and retaining people is expensive. Even though modern tools and practices may be more verbose and require more processing power than legacy code, using modern tools and practices is acceptable because the power of hardware has increased relative to its price. Buying more hardware is cheaper than hiring additional developers.

Another reason why components have become more important is the expanded use of networking and the Internet. Developers found that by breaking applications into components that could be deployed across multiple servers and connected by a network, they could get around the monolithic applications that didn't run.

Architecture evolution

The 1990s approach to component-based design was most often implemented in a physical, two-tier architecture: In this client/server architecture, some part of the application resides on the client, and another part resides on the server. A network connects the client to the server. In most cases, the data resided on the server, while the program to access the data resided on the client.

As time went on, developers became craftier at carving up their applications to run on multiple servers. This approach is generally referred to as *n-tier design* because an application can have any number, or *n* number, of tiers.

In reality, most applications stick with a two- or three-tier design. The standard for many applications is a logical three-tier design that looks like this:

- **Presentation layer:** The code for the graphical user interface
- **Business object layer:** The code that deals with the logic for your business domain, such as customers and orders
- **Data access layer:** The code that handles the process of moving data from the business objects into the data store, such as a database

One reason why this approach is popular is that it allows you to mix and match layers. Suppose that you create a business object layer that recognizes how your company manages its customers and orders. You can then create a Windows-based presentation layer and a Web-based presentation layer. Because all your business logic is specified only once in your business object layer, you have to maintain only one set of business objects. What's more, if you decide that users of smartphones need to be able to access your business object layer, you only have to create an additional presentation layer.

Contrast this approach with a monolithic application, in which you create separate applications for each user interface you need. You then have a Windows application, a Web-based application, and a third application for your smartphones. Each application has its own set of business logic for handling customers and orders. If you discover a bug or your business decides to changes its rules, you have to make changes in all three applications. With a tiered approach, the change is limited to your business object layer.

You still have to test your application before you roll it out. For example, you may need to modify the user interfaces so that they can handle a new field. Still, this technique is much preferred to having three sets of business logic.

You can implement this logical three-tier design in several ways in the physical world:

- **Two tiers:** In this typical client/server model, the presentation and business object layers might reside on the client, while the data access layer resides on the server.
- **Three tiers:** In this model, a different computer can host each layer.
- **More than three tiers:** If any of the layers is especially complex or requires significant processing power, each layer can further be divided and implemented on multiple computers.
- **No tiers:** If all the code resides on a single computer, it can be said to have one tier or no tiers.

Component management in .NET

The unit of deployment in .NET is an assembly. In an application with three tiers (physical, business, and data access), as described in the preceding section, each tier would likely have its own assembly. Of course, that's not to say you can have only one assembly per tier. Each tier can have as many assemblies as necessary to adequately organize the source code. The application may look something like Figure 3-1.

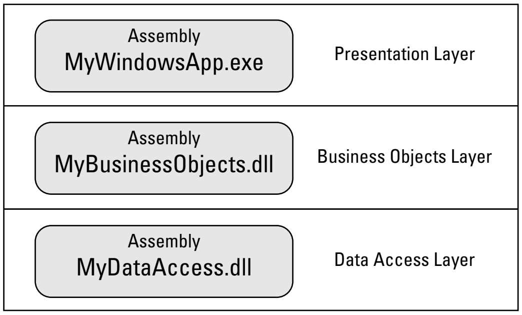


Figure 3-1: Each layer in a multi-tiered application is an assembly.

When an application is compiled, you can choose whether to compile it as an executable or a class library. In a three-tiered application, you might have the following assemblies:

✦ **MyWindowsApp.exe:** The file you would execute on a client PC. The file contains all the code for the presentation layer. The presentation layer consumes code from the other two layers by creating a reference to that code.

✦ **MyBusinessObjects.dll:** The class library that contains all the code for your business logic.

✦ **MyDataAccess.dll:** Another class library that contains all the code for getting data to and from the data store.

The reason that the business objects and data access layers are compiled as class libraries and not as executables is to prevent users from executing the files directly. Because no user interface is in those layers, users have no reason to execute them directly. The assemblies are just libraries of code that you can reference from other code.

To use the code that's in the business objects or data access class libraries, you create a reference in the executable file of the presentation layer, as shown in Figure 3-2.

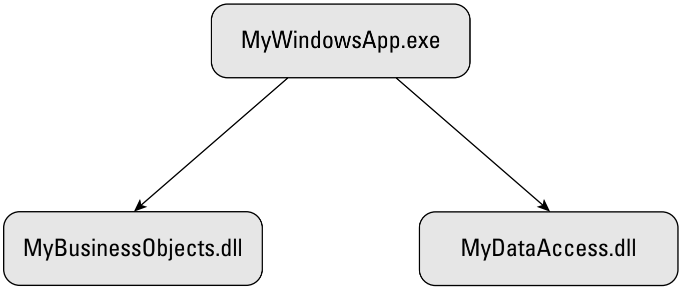


Figure 3-2: One assembly can use the code in another assembly by creating a reference.

After you create a reference to the assembly, you can access the code inside the assembly. Suppose that `MyBusinessObjects.dll` contains code to get a customer's address. To display the customer's address on-screen in your application, however, the presentation layer must reference the `MyBusinessObjects.dll` so that .NET knows where to get the code that gets the customer's address. See Book IV, Chapter 3 to read about referencing a class library.

After the reference is created, your presentation layer can access the code, which might look something like this:

```
MyApp.MyBusinessObjects.Customer customer;
customer =
  new MyApp.MyBusinessObjects.Customer("Smith, John");
MyDataList.DataSource = customer.GetCustomerAddress();
```

The first line of this code asks that the Customer class in the `MyBusinessObjects` class library set aside some memory for the customer Smith, John. The code that asks the data record for Smith, John from the database is stored in the `MyBusinessObjects` class library. Herein lies the beauty of assemblies: You can use this logic to retrieve the customer information in any of the presentation layers you create. Figure 3-3 demonstrates how the presentation layer accesses the code in `MyBusinessObjects.dll`.

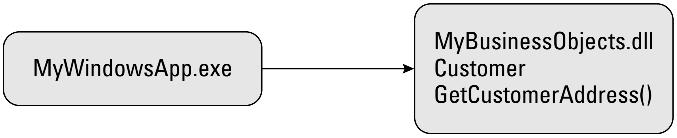


Figure 3-3: The presentation layer assembly can access the code in the business objects layer.

In the real world, you'd probably use a customer identification number rather than the customer's name. What if the database had two entries for Smith, John? You'd need some way to deal with the inevitability of having two or more different customers with the same name.

The second line of code tells .NET to use the customer name Smith, John from the preceding line to call the code that retrieves the customer address. The code then assigns whatever value is returned to a data list, which presumably is used to display the customer's address onscreen.

All the software that makes up the .NET Framework Class Library is made up of assemblies. Microsoft created a special place to store all the assemblies for .NET called the Global Assembly Cache, or GAC. (Yes, it's pronounced gack.) The GAC stores all versions of an assembly in a folder on your computer. You can store the assemblies you create in the GAC folder or in an application folder. See Book VI, Chapter 2 for more information on installing applications in the GAC.

What's in a namespace?

Notice that the code in the code sample is separated by dots, like this:

```
MyApp.MyBusinessObjects.
    Customer
```

Code is organized in .NET by using namespaces. Namespaces are a way to hierarchically organize code so that you can use duplicate names without creating confusion for .NET.

In this example, the top of the namespace hierarchy is named `MyApp`. Within the `MyApp` namespace is `MyBusinessObjects`. You might also find `MyWindowsApp` and `MyDataAccess` within the `MyApp` namespace. You access the code in those namespaces like this:

```
MyApp.MyWindowsApp
MyApp.MyWebApp
MyApp.MyDataAccess
```

Using namespaces this way allows someone else to have the same namespaces within a different namespace hierarchy. For example, if a coworker has the namespace `TeamApp`, the namespace might look like this:

```
TeamApp.MyWindowsApp
TeamApp.MyWebApp
TeamApp.MyDataAccess
```

In reality, you probably wouldn't prefix your namespaces with the word *My*. You'd most likely use your company's name or the name of the application. The namespace should allow you to easily identify the code contained within the namespace. Can you guess what kind of code you'd find in the `Microsoft.Word` or `Microsoft.Excel` namespaces?

Before the GAC came along, developers had no easy way to manage the versions of an application, so many people began to refer to the deployment of Windows applications as DLL hell. This term reflected the fact that the files that contain the code libraries — DLL files — could quickly become your worst enemy if you realized that you didn't have the right version of the file you needed in order to make your application work.

The term DLL hell was originally coined in an article entitled "The End of DLL Hell," which you can read at `http://msdn2.microsoft.com/en-us/library/ms811694.aspx`.

Managed Code Execution

One hallmark of modern software development is the way software is executed on a computer. Previously, software deployment and execution worked like this:

1. A programmer wrote source code, using some higher-level language, such as C# or Visual Basic.
2. Before the computer could understand the source code, it had to be *compiled,* or converted to the native language that the machine understands.
3. Because each kind of computer has its own native language, the code had to be compiled each time the programmer wanted to use it on a different kind of machine. There wasn't much compatibility between machines, so you often had to modify the existing source code to address the subtleties of the new hardware environment, as shown in Figure 3-4.
4. Because the code was converted to the native language of the machine, the code, when it was executed, had direct access to the operating system and the machine's hardware. This access not only made programming complex, but it also created an unstable operating environment.

The compiled .NET code that the CLR (Common Language Runtime) consumes is known as managed code. That's different than native code. *Native code* compiles into something that the CPU can directly use. *Managed code* must undergo the just-in-time (JIT) process by the CLR before the CPU can use it.

Back in the 1990s, the folks over at Sun had the bright idea of creating a programming language that could work on any machine. The Java Virtual Machine (JVM) that they built creates, on the hardware, a mini-environment that allows the software to be executed without touching the hardware, as shown in Figure 3-5. The JVM was smart enough to know how to make the software work with the hardware. Because of the resulting stable environment and simplified programming, Java was widely embraced.

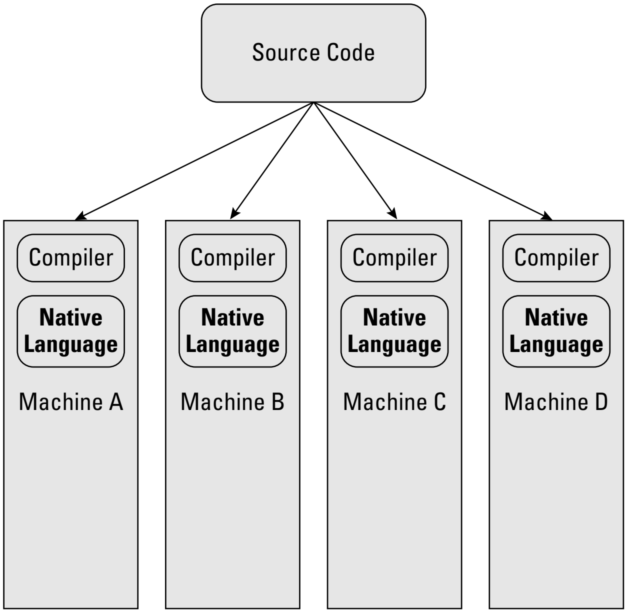


Figure 3-4: You have to modify software so that it works with each type of hardware platform.

The CLR virtual machine

Two big benefits of using the Common Language Runtime (CLR) virtual machine are that code is easily portable to multiple hardware platforms and the CLR provides a safer, more stable execution environment.

The CLR provides these additional services for managed code:

- Automatic memory management
- Verification that code will be executed as intended and hasn't been tampered with
- Assurance that managed code can interact with unmanaged code (code running outside the CLR)
- Support for multiple versions of the code

See the nearby section "Taking Out the Garbage" for more information on automatic memory management.

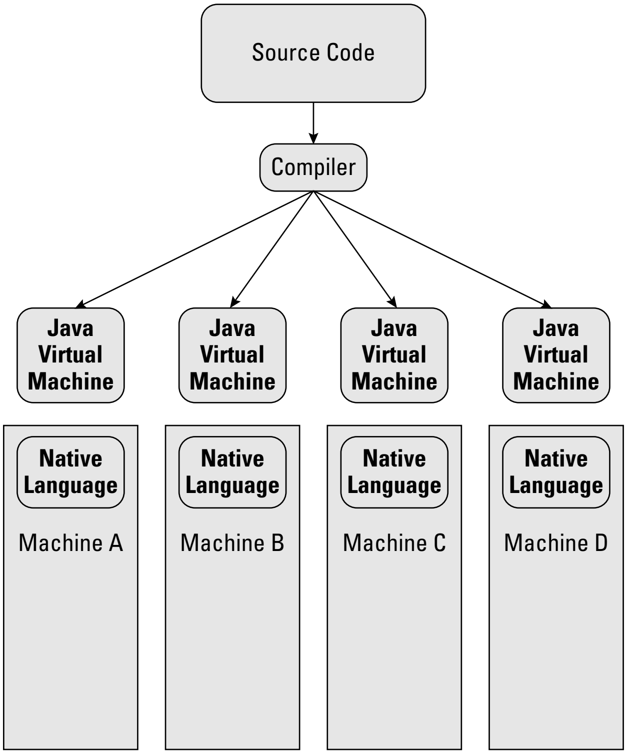


Figure 3-5: The Java Virtual Machine creates a mini-environment for running software.

Ever mindful of its competition, Microsoft sneaked a peek at the Sun JVM and liked what it saw. Over time, all the different Microsoft teams that were working on building development tools and languages started coalescing to work on their implementation of the virtual machine. The Microsoft .NET Framework was born.

The Microsoft version of the virtual machine is slightly different in implementation from Java, although they're similar conceptually. The virtual machine in .NET is the CLR. These steps show how using virtual machines changed the deployment and execution of software:

1. **A programmer writes source code, using a higher-level language, such as Visual Basic .NET.**
2. **The VB.NET compiler converts the source code into Microsoft Intermediate Language (MSIL), which is the native language of the CLR virtual machine. Rather than convert the source code into the**

hardware's native machine language, the compiler creates intermediate code that the CLR understands. Source code compiled into MSIL is managed code.

All language compilers that target the .NET CLR convert the source code into MSIL. Programmers can develop the source code using multiple programming languages, and the compiler can compile this code to create a single application.

3. **The compiler also creates metadata about the source code. The metadata identifies to the CLR all the assemblies and other files that the source code needs in order to be executed properly. The CLR is responsible for resolving dependencies before the code is executed.**

4. **When the code's assembly is executed, it's compiled a second time from the MSIL into the native code for the hardware platform on which the code is being executed. This second compiler is the just-in-time (JIT) compiler because the MSIL code is compiled just before it's executed.**

   When code is compiled by using the JIT compiler, the compiled code is stored in memory. If the code is used again, it doesn't have to be compiled again. Rather, the copy of the code that's in memory is used. When the execution is complete, the code is removed from memory. If the code is called again after being removed from memory, it must be compiled into native machine code again by the JIT compiler.

5. **Managed code (code that runs in the CLR) is executed within the CLR virtual machine. As a result, the code doesn't have direct access to the hardware and operating system. The code is more stable and less likely to crash the system. To read more about the benefits of running code in the CLR virtual machine, see the sidebar "The CLR virtual machine."**

Figure 3-6 diagrams how code is executed by using the CLR.

The architecture used by the JVM and Microsoft .NET is part of an international standard for how software execution should be managed. The languages of .NET are expected to conform to the Common Language Specification. The CLR of .NET is the Microsoft implementation of the Common Language Infrastructure.

Taking Out the Garbage

One big service of the .NET Framework is *automatic memory management,* or garbage collection. Microsoft didn't make up the term *garbage collection* — it's an authentic computer science term that applies to computer memory.

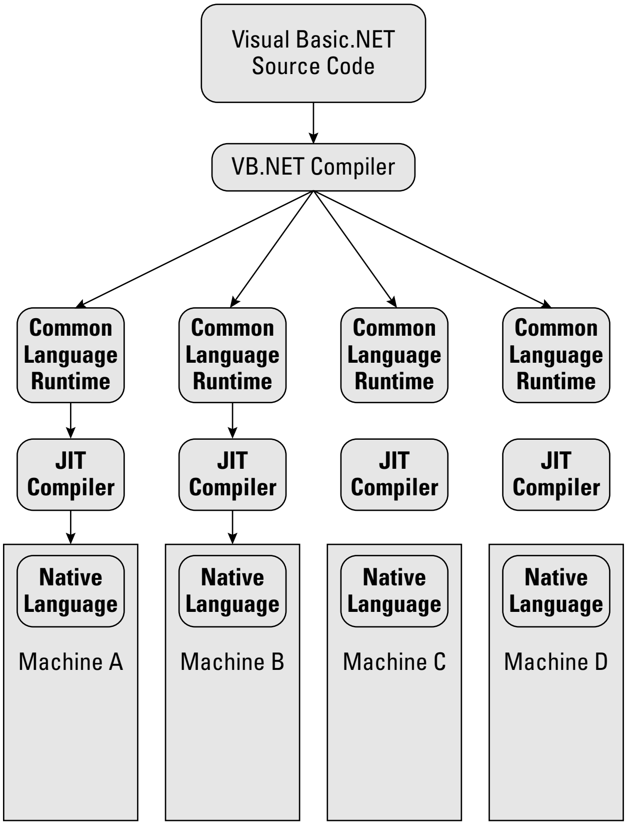


Figure 3-6: The CLR is responsible for executing .NET code.

Before a computer program can use a resource, such as a database, the program must request that the computer set aside memory for the resource. After the program stops using the memory, the data stored in the memory is referred to as garbage because it's no longer useful. To reclaim the memory for another use, the CLR determines when it's no longer in use and disposes of it.

A garbage collector, such as the one in the .NET Framework, takes care of the business of asking for memory and then reclaiming it after the program is done using it. The garbage collector collects the garbage memory and then makes it available for use again.

Memory is used to store the resources a program needs to complete tasks, such as

- Storing data, such as customer and order records
- Displaying information on the screen or sending it to a printer

- Opening network connections for sending data
- Opening a database connection
- Opening a text file

The garbage collector frees the developer from having to perform these tasks:

- Tracking how much memory is available and how much is used
- Assigning memory to program resources
- Releasing memory when resources are done using it

The garbage collector allows developers who are creating a program to focus on customers, orders, and other business domain entities, rather than on memory management. Using a garbage collector eliminates two common errors that developers make when they have to manage their own memory:

- **Forgetting to free memory:** When a program doesn't release a computer's memory after the program's done using it, the memory is quickly filled with garbage and can't be used for useful tasks. This error is often called a memory leak. When a program runs out of memory, it crashes. What's worse is that memory leaks can cause other programs — and even your whole computer — to crash.
- **Trying to use memory that has already been freed:** Another common error that developers make is trying to access a resource that has already been removed from memory. This situation can also cause a program to crash unless the developer makes the program test whether the resource is still available. A developer who forgets that the resource is freed probably won't remember to test the resource before trying to use it.

Allocating memory

To prevent you from having to manage the memory yourself, the CLR allocates memory for program resources this way:

1. **The CLR reserves a block of memory, which is the managed heap.**
2. **When you ask for resources, such as to open network connections or files, the CLR allocates memory for the resource from the managed heap.**
3. **The CLR notes that the block of memory is full so that it knows to use the next block of memory when you ask for a resource the next time.**

Although you still have to tell the CLR that you need the resource, all the details about which memory addresses are occupied, as well as what they're occupied with, are handled by the CLR on your behalf.

Releasing memory

This section describes how the CLR knows when to release the memory.

The CLR divides the memory in the managed heap logically into manageable pieces, or *generations.* The garbage collector (GC) uses three generations: 0, 1, and 2. The GC operates under the assumption that newly created objects will have shorter life spans than older objects.

Whenever you ask for a new resource, memory is allocated from the portion of the memory block designated as Generation 0. When Generation 0 runs out of memory, the GC starts a collection process that examines the resources in memory and frees anything it deems as garbage (anything that's unusable to the application). After freeing as much memory as possible, the GC compacts the memory so that all the remaining resources are placed next to each other in a neat stack. Compacting memory is something the developer doesn't have to do.

How do resources get into Generations 1 and 2? If the GC can't reclaim enough memory from Generation 0 to accommodate your new requests, it tries to move the resources from Generation 0 to Generation 1. If Generation 1 is full, the GC tries to move them to Generation 2. This process is how older resources that are still in use graduate through the generations.

REMEMBER

Garbage collection isn't something you should try to control—it happens automatically.

The GC doesn't know what to do with certain kinds of resources after it realizes that you aren't using them any more — for example, resources from the operating system when you work with files, windows, or network connections. .NET provides you with special methods — named `Close`, `Finalize`, and `Dispose` — that you can use to tell the GC what to do with these resources.

The vast majority of resources you use in .NET don't require you to release them. If you're unsure whether you should make method calls before you're completely done with an object, go ahead and call the `Close`, `Finalize`, or `Dispose` method while the object is still in scope. See Book V, Chapter 3 to get the skinny on how to use methods such as `Finalize`.

Garbage collection isn't unique to .NET. Java also provides garbage-collection services to its developers. Read the article that describes how garbage collection works in Java at `www.javaworld.com/javaworld/jw-08-1996/jw-08-gc.html`.

You'll find that the process is remarkably similar to garbage collection in .NET.

Chapter 4: The Languages of .NET

In This Chapter

- **Introducing the programming languages of .NET**
- **Finding the language that's right for you**
- **Figuring out the fundamentals of programming**

Part of the draw to Visual Studio 2010 is that it has many tools that make software development a breeze. You can fire up Visual Studio 2010 and, in no time flat, have a Windows or Web application that looks very professional.

This chapter introduces the programming languages you're most likely to use.

More than Just a Pretty Face: A Windows Application

Take a look at the application shown in Figure 4-1. It looks like any other Windows application you use. I created it in about five minutes by using Visual Studio 2010. Even though it uses only two lines of added code, it looks like an element from a full-blown Windows application.

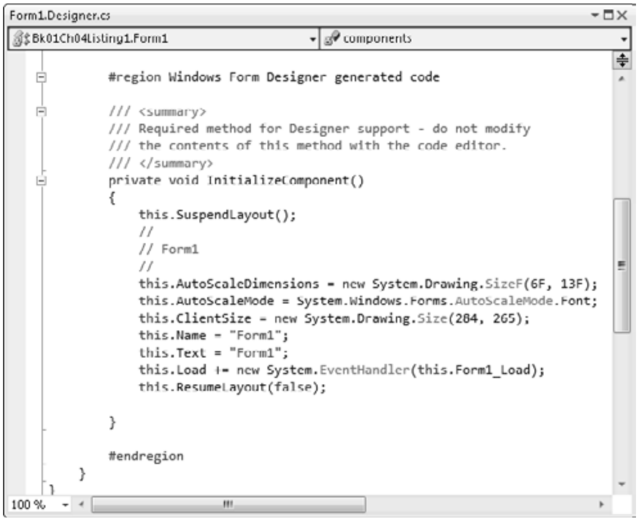
Figure 4-1: Create professional-looking Windows applications by using Visual Studio.

ON THE WEB

You can download this simple project from this book's companion Web site at `www.dummies.com/go/vs2010`.

Visual Studio provided the Windows Form shown in Figure 4-1, and you just drag and drop the text boxes and buttons onto it. Behind the scenes, Visual

Studio takes care of all the programming necessary to make this screen work by using code generation, which writes all the necessary code for you. All you have to do is use a designer (a graphical tool) to paint the screens and make them look the way you want. You can also use designers to generate code to build everything from reports to database models in your applications. Figure 4-2 shows an example of code generation.



```
        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.SuspendLayout();
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(284, 265);
            this.Name = "Form1";
            this.Text = "Form1";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.ResumeLayout(false);

        }

        #endregion
    }
}
```

Figure 4-2: Visual Studio works behind the scenes to generate the code to make an application work.

At some point, however, you need your application to do more than just look pretty. When someone clicks a button or a menu item, your application should respond. This response takes place only when you write the code that tells the application what you want it to do, and that, my friend, requires programming.

Looking at the Languages of .NET

Even though Visual Studio can do a great deal for you, it can't write your program for you. When someone clicks a button, you have to provide code that carries out the desired action. The following list describes the kind of code you have to create to perform a specific task:

- ✦ **Open a new window and display it to the user.** You have to write the code to open the new window.
- ✦ **Save a record to the database that the user entered.** You have to write the code to save the record to the database.
- ✦ **Perform a calculation based on data that the user entered.** You have to — you guessed it — write the code to perform the calculation.

Programming languages have come a long way since they were first created. Modern programming languages are higher-level languages than their predecessors because each line of code that you write gets translated into multiple lines of instructions for the computer.

Before the introduction of higher-level languages and after the era of punch cards and programming in hex codes, programmers wrote programs by using assembly language. Assembly language matches computer instructions line for line; in other words, each line of assembly language that you write is equal to one line of computer instructions. Writing programs in assembly languages is time-consuming and requires extensive knowledge of how computers work.

The need for higher-level languages that allow programmers to write one line of code that's translated into multiple lines of computer instructions had to be addressed. These higher-level languages allow developers to be more productive. Also, programmers don't have to be as knowledgeable about the inner workings of computers.

The languages of .NET are all examples of higher-level languages. Microsoft has four default languages for .NET:

- Visual Basic
- Visual C#
- Visual C++
- F#

Not all higher-level languages are created equal. Each of the languages of Microsoft .NET has its own set of advantages and disadvantages. The next few sections describe each of these languages, starting with the old standby, Visual Basic.

Visual Basic

In the area of Windows programming, Visual Basic has been around for quite awhile. This language has many devotees, and programmers love it because it's easy to learn, write, and debug. You can easily begin writing workable software very quickly.

You can use Visual Basic to build these types of applications:

- For Windows
- For the Web
- For mobile devices

To make Visual Basic work with the Microsoft .NET Framework, the language had to be totally rewritten. In the process, many long-time Visual Basic programmers felt that Microsoft betrayed the defining characteristic of Visual Basic: It's easy to use.

Even though Visual Basic.NET was introduced in 2002 (although the first beta was released in 2000), many new programs are still written in Visual Basic 6, the last version of VB before it was .NET-ized. Microsoft is aware that many developers still haven't made the move to Visual Basic.NET, so the company included a significant amount of documentation to help Visual Basic 6 developers take the plunge. You can find more information at `http://msdn.microsoft.com/en-us/kehz1dz1.aspx`.

Even though the process of moving from Visual Basic 6 to VB.NET has been painful, it places VB on the same playing field as all the other .NET programming languages. Had Microsoft chosen not to rewrite Visual Basic, VB may have been on its way to language obscurity.

Rewriting Visual Basic made the language more powerful by turning it into a true object-oriented language. VB.NET allowed Microsoft to implement

- Customer-requested changes, such as inheritance and threading
- Complete access to all the features of .NET
- Interoperability between Visual Basic and the other .NET languages
- The removal of features such as `GoSub/Return` and `DefInt`
- Consistency between Visual Basic and other languages and language standards

This Hello World code sample from MSDN was written in Visual Basic.NET:

```
' A "Hello World!" program in VB.NET
Module Hello
  Sub Main()
      MsgBox("Hello, World!") ' Display message
  End Sub
End Module
```

Notice how the syntax is simple. Neither extra braces nor punctuation are required. For more information about this code sample, see MSDN at `http://msdn.microsoft.com/en-us/3cf7t4xt.aspx`.

Visual C#

Released in 2002 (although the beta was released in 2000) with the first version of the Microsoft .NET Framework, Visual C# was created especially to take advantage of Microsoft .NET features. C# has many characteristics in common with other languages in the C family of languages:

- All statements end with a semicolon.
- Blocks of code are enclosed in curly braces.
- The language is case sensitive.

Check out the C# version of Hello World:

```
using System;
// A "Hello World!" program in C#
namespace HelloWorld
{
    class Hello
    {
        static void Main()
        {
            System.Console.WriteLine("Hello World!");
        }
    }
}
```

The C# version of Hello World requires a few more lines than Visual Basic. Some (myself included) find the braces a nice visual cue for grouping code together. Other developers see them as a waste of space. For more information on this code sample, see the MSDN documentation at `http://msdn2.microsoft.com/k1sx6ed2.aspx`.

The C# language is useful for writing object-oriented applications. In addition to showcasing the .NET Framework, Microsoft created C# to leverage the large numbers of Java, C, and C++ programmers who can easily learn the language. People with a strong C++ and Java background have learned C# in as little as two weeks.

In its attempt to lure Java developers, Microsoft also created the Java Language Conversion Assistant (JLCA), which converts Java source code into C#. This tool is good for converting fairly simple programs. Graphical conversions don't work, though.

Visual C++

Visual C++, the most complex language in the Microsoft .NET offerings, is used to develop systems-level software, such as graphical user interfaces (GUIs), device drivers, and mobile applications. Visual C++ is a development environment for creating Windows applications by using the C++ programming language. Visual C++ can create applications that produce native code, or applications that produce managed code.

Here's a C++ version of Hello World:

```
#include <iostream>
// A "Hello World!" program in C++
using  namespace std;
int main()
{
   cout << "Hello World! \n";
   return 0;
}
```

The C++ syntax is a bit more cryptic than other high-level languages, such as VB or C#. For example, `cout` is short for console output. C++ syntax isn't as intuitive as other languages.

You can find a simple Hello World example at `http://msdn.microsoft.com/en-us/ms386445%28VS.71%29.aspx`.

TIP

Check out C++ For Dummies, 6th Edition, by Stephen Randy Davis (Wiley), if you're interested in creating C++ applications. You have to get your mind around C++ before you can get any benefit from Visual C++.

You might think of Visual C++ as a development environment inside a development environment. Visual C++ allows you to use C++ syntax to write programs. Visual C++ provides several tools that shortcut the amount of effort required to create a standard C++ application, such as

- **Compilers:** Target the .NET Common Language Runtime (CLR) and the x86, x64, and Itanium hardware
- **Libraries:** Include Active Template Library (ATL), Microsoft Foundation Classes (MFC), standard C++, C Run-Time (CRT), and the C++ Support Library to support CLR programs
- **A development environment:** Supports project management and configuration; source code editing and browsing; and debugging

You can use Visual C++ as part of the Visual Studio IDE, or you can use its individual tools on the command line. Although C++ is a powerful language, it's also the most difficult of all the .NET languages to learn.

F#

Microsoft developers have been working on the F# language for a number of years. They actually released an early version in 2008, but it's now a core language in Visual Studio 2010.

Here’s an F# code sample of Hello World:

```
open System
let _ =
    Console.WriteLine("Hello F# World!");
```

F# syntax uses method class similarly to C#. Note that it has semicolons at the end of lines that call methods.

There is a more complete description and explanation of using F# in Chapter 1 of Book III. I also have a short F# tutorial on the Web site that you can get a link to from www.dummies.com/go/vs2010.

Choosing the Right Language for the Job

One key feature of Microsoft .NET is that in order for a programming language to work with .NET, its compiler must be able to translate the language into MSIL. Because you can convert all the languages to MSIL, all instructions that get executed by the computer should be similar, if not the same.

In other words, the computer doesn’t care what programming language you choose when you’re writing programs for .NET. MSIL equalizes all the languages as far as the computer is concerned.

Choosing a language should be motivated by human needs, not by the computer. Think about what you’re trying to accomplish when you’re making a decision. Your goal should be to write programs that do what you want them to do and that your teammates can read and maintain.

As you make your decision, keep these considerations in mind:

- ✦ Microsoft has positioned Visual Basic as an easy entry point into Microsoft .NET. If you’re new to programming or new to .NET, Visual Basic is probably the way to go.
- ✦ If you have experience with using Java, C, C++, or even one of the scripting languages, such as Perl or Python, you might appreciate Visual C#. C# syntax is similar to these C-based languages.
- ✦ If you’re primarily concerned about performance, you probably should choose unmanaged C++. If you choose unmanaged C++, then you should look into modern libraries, such as the Standard Template Library (STL) and Boost (www.boost.org) for your development. STL provides highly-optimized collections, such as sets, maps, lists, and hash tables. Boost contains expertly designed, peer-reviewed libraries that provide services such as network communications, threads and thread

pools, functional programming and lambda functions, reference-counted smart pointers, and much more. Be aware, however, that these libraries require significant knowledge of C++, but the C++ programmer aids your development by providing many of the same services as the .NET framework, but with the added performance of unmanaged code.

WARNING!

Unless you have the formal training and practical experience to write tight code (code that doesn't waste any memory or processor cycles), you probably won't gain any performance edge by using C++ without .NET. Don't buy the hype that you can write faster programs with C++. Although that's possible, most programmers don't have the experience to do it. Furthermore, in the amount of time you spend trying to figure it out, you could already have the program finished in Visual Basic or C#.

If you ever build Web applications with ASP.NET, you already use JavaScript. Many of the ASP.NET server controls generate JavaScript to make them work. You should also try JavaScript whenever you need to get something done in a jiffy, such as when you parse a text file. You might find that you like the laid-back style of JavaScript better than its more buttoned-up brethren.

For a Microsoft comparison of the languages of .NET, see `http://msdn.microsoft.com/en-us/czz35az4.aspx`.

Most people are choosing between Visual Basic and C#. Your historical use of one language over the other often dictates your choice. People who use Visual Basic tend to be less interested in aesthetics and more interested in getting the task done so that they can move on. C# programmers often speak of the purity and expressiveness of the language.

AJAX attacks!

An exciting development for .NET Web development is its new Web development methodology: AJAX (Asynchronous JavaScript And XML). It uses JavaScript and other technologies related to the Web to provide Windows-like user interfaces to Web pages. JavaScript is no longer just for accessing client-side content. With AJAX, you can use JavaScript to interact with the server, too.

Getting all the AJAX technologies to work together to create this rich Web experience is a daunting challenge. The Microsoft initiative named Atlas intends to provide a toolbox of Ajax controls. You can start using Atlas right now. Check out the Microsoft Atlas page for more information about Atlas and JavaScript: `www.asp.net/ajax/default.aspx`.

Having used both, I offer these highlights:

- Visual Basic is easy and fast to learn, but it's more verbose.
- C# lends itself well to object-oriented development, and far more code examples are available than for VB.

Typically, ASP.NET code has two parts. The first is the presentation page with the HTML-like objects. The second is the code that responds to page events, known as code-behind. I like to use C# for all my ASP.NET code-behind files, as well as use C# for all my business objects.

Becoming a Good Programmer

Learning how to program is probably more important than choosing a language. By learning how to program, you can move fluidly among languages.

All programming languages share common themes, or programming constructs. You should become familiar with some of the common programming constructs:

- Declaring variables
- Understanding variable types
- Knowing when and how to use variables for loops
- Controlling the flow of your program by using conditional `If...Then` statements
- Using arrays to store a series of data, such as the months in a year

After you finish this book and have the skills, you can get more advanced information. If you're interested in moving beyond this book, start with the course catalog for the Computer Science department at your local college or university. You're more likely to find "what it means to program a computer" courses that cover programming fundamentals. Be careful about applied technology courses or books that teach a specific language, such as Visual Basic or Java. If possible, take a look at the syllabus or talk to the professor or lecturer before signing up. Explain that you want to get more out of the course than just language syntax.

Building software involves more than just writing code, and software development consists of several phases. This book shows you how to use Visual Studio 2010 in many of these different phases.

After the requirements gatherers gather, the analysts analyze, and the designers design, at some point, somebody has to write some code. If you want that person to be you, you need to learn language syntax, or the grammar of a programming language.

Learn the syntax

If you want to learn language syntax, consider not starting with Visual Studio because Visual Studio doesn't possess any special "magic" for creating applications. Visual Studio lets you develop software at a higher level of abstraction. As a result, Visual Studio generates a lot of code for you — code that you probably won't bother to take the time to understand. To be frank, the code that Visual Studio generates probably isn't the code you want to learn from.

REMEMBER

Writing cryptic code isn't the goal. Rather, the goal is to write code that's easy to read and maintain. The code generated by Visual Studio can be complex and abstract. It's good code because it's highly reusable, but it can be hard to understand.

Some people argue that you shouldn't bother trying to memorize syntax. As long as you know the basic programming constructs, you should be able to program in any language. Don't forget that companies want developers with high levels of productivity, which usually translates into using some kind of tool, such as Visual Studio.

Write your own code

When I was in college, the recommended order for taking courses was to start by using a text editor and manual compiler in a language like C; learn a C-derivative scripting language, such as JavaScript or Perl; and then learn Visual Basic .NET with Visual Studio.

Many of my classmates, however, chose instead to start with the VB.NET course and work their way backward. Unfortunately, they started with the easiest way to develop software. When they got to the harder courses, where you had to manage your own memory and create your own data structures, they were lost.

I had the same experience with Structured Query Language (SQL), which is the language of databases. When I first started working in information technology, I was doing a lot of report design, mostly relying on report-design software to graphically build my database queries. Report designers, such as Crystal Reports, generate the SQL necessary to pull the data out of the database.

When I started doing application development, I realized quickly that my SQL skills were deficient. I immediately started making myself write out all queries by using a native SQL tool. After I stopped relying on code-generation tools and started writing my own SQL, I quickly learned the language. Now

I get to choose whether using a code-generation tool is the best way to go, given the task that I need to accomplish.

Here's the bottom line: You need to know some language syntax if you expect to become more than a copy-and-paste coder. Here are some tasks you can undertake to become a better programmer:

- Learn the principles of object-oriented design and programming.
- Learn the basic programming constructs that are common across all programming languages.
- Pick a language and learn how that language implements object-oriented programming and basic programming constructs.
- If you know that you'll work with a specific technology, such as Windows Forms or Web services, commit yourself to learning as much as possible about that technology and its best practices development.
- Practice, practice, practice.

If you invest in these techniques, your code will be well written, no matter what.

Spend some time in other language studies.

TIP

If you want to separate yourself from the development pack, pick a language and spend four consecutive weekends working with that language outside the Visual Studio environment. This technique is also a good way to teach yourself a new language, such as Java.

You might also need to familiarize yourself with some of these other languages and technologies:

- **Standards for Web development,** such as HTML, Cascading Style Sheets (CSS), JavaScript, and AJAX, to help with troubleshooting
- **File formats,** such as XML and really simple syndication (RSS), so that you can transfer data between disparate systems
- **Query syntax** of SQL, for writing stored procedures and queries for reading and updating databases
- **Regular expressions,** for manipulating text
- **Windows Active Directory and security,** for developing applications that can automate tasks on a network and perform tasks, such as user authentication
- **Object models for Office applications,** such as Word and Excel, and server applications, such as SharePoint and BizTalk

Chapter 5: The Many Editions of Visual Studio 2010

In This Chapter

- **Sorting out multiple editions of Visual Studio**
- **Deciding which edition is right for you**
- **Choosing the right MSDN subscription**
- **Getting acquainted with Visual Studio Team System**

Sorting out all the different editions of Visual Studio is mind-numbing. Microsoft provides a Visual Studio edition for developers at all levels of experience, from absolute beginners working on their own to large-scale development teams with thousands of members.

This chapter helps you make sense of the different editions of Visual Studio so that you can decide which one's right for you. In this chapter, you discover the features of Visual Studio Team System and read about the future of Visual Studio.

Making Sense of the Visual Studio Editions

Although Microsoft still offers the stalwart Professional Edition, the company expanded its offerings to reach out to everyone from novices all the way to high-end, large-scale development teams.

Visual Studio offers these editions, listed in order from the simplest to most complex features included in the edition:

- Visual Studio Express Editions
- Visual Studio Professional
- Visual Studio Team System

One of the major goals of Microsoft with Visual Studio is to enable personal productivity for all software developers. This list describes which edition you should use if you're in one of these groups:

- **You're a hobbyist, student, or novice.** Check out the Express Editions. You should also consider the Express Editions if you don't know whether to upgrade from a previous version of Visual Studio.
- **You do occasional development work by yourself.** You should appreciate the get-down-to-business features of Visual Studio Professional Edition.
- **You create applications that use Office as their user interface (UI).** Visual Studio Professional Edition makes creating Office applications dead simple.
- **You're a professional software developer who works alone or on a small team.** Visual Studio Professional is geared toward you.
- **You're part of a larger development team.** Your team should check out Visual Studio Team System.

Visual Studio Express Editions

The Express Editions of Visual Studio, which are less feature-rich than the other editions, are perfect for students, hobbyists, and folks just getting started. Rather than have one large, overwhelming soup-to-nuts software development tool, the Express Editions come in many flavors, depending on what type of development you want to do:

Web: Create Web applications and Web services by using Visual Web Developer 2010 Express. This edition enables you to write code by using C# or Visual Basic and provides editors for HTML and XML. It has a built-in Web development server that you can use to build and test your Web application. Numerous starter kits are provided to help you jump-start your projects.

Database: Most applications are data-centric. SQL Server 2008 Express Edition is a fully functional database management system that integrates with Visual Studio. You can use Visual Studio to create databases, write stored procedures, and add data. Although database sizes are limited to 4GB, the databases are compatible with the Enterprise version of SQL Server 2008. So, the skills you acquire are transferable to SQL Server 2005 and SQL Server 2008.

Windows: Express Editions for Windows development includes editions for Visual Basic, C#, C++, and F#. Use Visual Basic and C# for building Windows applications. Use Visual Basic, C#, C++, and F# for building console applications and reusable components. You can use C++ to create both managed and unmanaged code. To build Web applications, use the Web Developer Edition.

Visual Studio Tools for Office

Visual Studio Tools for Office (VSTO) is no longer a separate edition, as it was with Visual Studio 2005. Although VS Tools for Office is now part of Visual Studio 2010, it's worth talking about separately so that you can appreciate all that it brings to the table. VSTO (pronounced vis-toe) is geared toward independent software vendors and systems integrators who build custom Windows and Web applications for the Microsoft Office system.

With VSTO, you get these features:

- Visual Basic and C# programming languages
- Integrated visual designers for working with databases
- Advanced debugging tools
- Designers geared toward building applications that use Word, Excel, and InfoPath as their UI

Most beginners will find that they make few compromises by using Visual Studio Express Editions. You get these benefits:

- Your developed applications are fully functional. They aren't watered-down versions.
- As your needs grow, you can scale up to one of the higher editions.
- Your skills are transferable to other Visual Studio editions.
- The price is right. You can download all the editions for Web and Windows development for free from the Microsoft Web site. Microsoft originally planned to charge $49 for the products after offering them free for one year, but it decided to make the products free permanently. SQL Server 2008 Express Edition is also available at no charge.

Download these tools at `www.microsoft.com/express/downloads`.

The Express Editions of Visual Studio are an excellent choice for students or anyone wanting to learn how to program. These choices weren't always available: As a result, students used tools like Microsoft Access, which is worlds away from enterprise-level database management tools, such as SQL Server. Kudos to Microsoft for providing these tools and making them affordable.

Visual Studio Professional

Visual Studio Professional is geared toward professional .NET developers. This edition provides even more advanced visual designers and a tightly integrated toolset. If you develop software for a living, this edition is the one you're most likely to use.

Visual Studio Professional provides these features:

- All the Microsoft .NET programming languages
- Integrated visual database tools
- Designers for XML schema design and XML style sheets
- Advanced debugging and deployment tools
- The ability to develop stored procedures, functions, and triggers for SQL Server 2005 or SQL Server 2008
- Crystal Reports

Visual Studio Team System

The Visual Studio Team System (VSTS) has created a lot of buzz. VSTS is important because

- It acknowledges the specialized roles that people play on large development teams.
- It's the Microsoft attempt to move into the tools space of the software development life cycle.

VSTS isn't a product; rather, it comprises four client editions of Visual Studio:

- Visual Studio for Architects
- Visual Studio for Database Professionals
- Visual Studio for Software Developers
- Visual Studio for Testers

The Team System also includes a server component — the Team Foundation Server (TFS) — that enables collaboration among the client editions of VSTS.

Because VSTS offers many more features than most versions of Visual Studio, this chapter digs into it in more detail in the upcoming section "Developing When It's a Team Effort."

Choosing the Right Edition

With numerous editions of Visual Studio and numerous features strewn across these editions, finding an edition that's right for you can seem daunting. The answer almost always comes down to money (somebody else's, you hope).

Because pricing can change, I don't quote specific prices here. However, here's a ballpark idea of how much these editions cost:

- **Express Editions:** Free
- **Professional Edition :** $799
- **Professional Edition with MSDN**: $1,199 new/$799 renewal
- **Premium Edition with MSDN:** $5,469 new/$2,299 renewal
- **Ultimate Edition with MSDN:** $11,924 new/$3,814 renewal
- **Team Foundation Server:** At the time of this writing the pricing is not available.

The price you pay depends on whether you're upgrading or paying full retail price.

Helping you decide whether to choose the Team System is beyond the scope of this book. Chances are that your employer will make the decision. Nevertheless, as a .NET developer, you have a responsibility to understand the role of the Visual Studio Team System. Check out the later section "Developing When It's a Team Effort" for more information.

Subscribing to the Microsoft Developer Network

MSDN provides you with access to Microsoft operating systems, server software, and productivity applications for development purposes.

You can choose from three levels of MSDN subscriptions:

- **Academic:** For about $800, you have access to all the Microsoft operating systems, such as Windows Server 2008 and Virtual PC.
- **Professional:** For about $1,200, you get Visual Studio 2010 Professional Edition, SQL Server 2008, SQL Reporting Services Developer Editions, and Virtual PC, in addition to access to Microsoft operating systems.
- **Premium:** This provides you with the Virtual PC, SQL Server 2008 Developer Edition, and Visual SourceSafe operating systems; server products, such as BizTalk and SharePoint; and Microsoft Business Solutions software, such as Great Plains and Microsoft CRM. You also get to choose an edition of Visual Studio. The price you pay for a subscription depends on the edition of Visual Studio you choose:
  - **Visual Studio Professional:** This costs about $2,500.
  - **Visual Studio Team Edition for Software Architects, Testers, or Software Developers:** Choose one of these editions for about $5,500.
  - **Visual Studio Team Suite:** For about $11,000, you get access to all three Team Edition editions of Visual Studio.

The Visual Studio Team Editions and Team Suite with MSDN Premium Subscriptions include the TFS Workgroup Edition with five client-access licenses. The full-blown TFS product requires an additional server license.

Check out the latest pricing at `http://msdn.microsoft.com/vstudio/howtobuy`.

For more details about what you get with each level of MSDN subscription, go to `http://msdn.microsoft.com/en-us/subscriptions/default.aspx`.

MSDN subscriptions provide you with development and testing licenses only. You can't use the software you download from MSDN in a production environment. You can only test or develop.

A single MSDN subscription provides you with ten licenses for each product you download. You may install the software on ten computers one time or ten times on one computer. Only individuals with MSDN subscriptions may use the software. In other words, each member of your team needs an MSDN subscription if they plan to access software downloaded from MSDN. The licenses are valid forever, even if you choose not to renew your MSDN subscription. You may continue using the software for testing and development after your MSDN subscription expires.

Each developer requires a separate MSDN subscription. Just because a single MSDN subscription has ten licenses doesn't mean ten people can use it. Each person must have his own subscription.

Weighing your options

After you have an idea of what the pricing structure looks like and what you can get for your money, look at these scenarios to help you decide which edition of Visual Studio 2010 is right for you:

- **If you want to save some money or mentor a young programmer:** Be sure to download all the Express Editions for free. Even if you think that you'll never need them, download them now at `www.microsoft.com/express/downloads`.

  You never know when you might need a free edition of a relatively powerful development tool.

- **If you're a college student:** Check with your school bookstore or your department to see whether your school has an academic license with Microsoft. When I was in college, I could pick up a copy of Visual Studio Professional for $25 — only $5 per disc. Your school may have a similar license.

- **If you work in a small- to medium-size business or have your own consulting business:** Choose Professional Edition if you want to use Crystal

Reports Designer to create reports. If you don't mind having separate tools for Web-based and Windows-based development, you can save a few bucks and go with the Express Editions. If you intend to do more development work, choose the Professional Edition. You benefit from using Class Designer and being able to access remote databases.

- **If you do extensive development work with Office:** Visual Tools for Office is a no-brainer — except, of course, that you get all the VSTO tools with the Visual Studio Team Editions. If you do only occasional Office work, stick with Professional.

Consider these scenarios as you decide whether to choose MSDN:

- **You're a professional developer or have developers working for you.** Spring for the MSDN subscriptions. Consider it part of the cost of doing business. Most developers expect to have access to these tools.
- **You're a one-person show or you're on a budget, so maybe you can get around buying an MSDN subscription.** In that case, buy a copy of Virtual PC or VMWare and then install evaluation versions of whatever software you need to work with. You can save images of the evaluation versions if you need to reuse them for other projects.
- **As you make your decision about which tools to use, remember that Microsoft has a profit motive for selling Visual Studio.** Why is it giving away the Express Editions? The answer is that all marketers know that a trial offer is the best way to sell something.

Evaluating your alternatives

Over the years, I have worked in a number of Microsoft shops. A Microsoft development shop tends to go with "all things Microsoft" and not even consider alternatives. Microsoft has many useful products, but they're not always the right choices for all situations. They're often the easiest choices, though, especially if your infrastructure and development efforts are heavily invested in all things Microsoft.

Before you commit yourself and spend huge sums of money, ponder some alternatives:

- Supplement Visual Studio Professional with open source tools, such as NUnit for unit testing and FxCop for static code analysis.
- Consider using Project Mono, an open source version of C#.
- Explore alternative development environments, such as Eclipse and NetBeans.

Nothing about using Visual Studio is inherently bad. Just remember that when you frame solutions by using Microsoft tools, or any vendor's tools, you narrow your choices and lose perspective. By evaluating alternatives, you can see whether Visual Studio is the way for you to go.

TIP

When you decide to go with Visual Studio, you're not only getting a productivity tool, but you're also joining a community. Using Visual Studio and Microsoft .NET will no doubt shape how you approach solutions development.

Developing When It's a Team Effort

With Visual Studio 2010, Microsoft continues the move that Visual Studio 2005 started and moves into the realm of software development life cycle tools in a big way. The products that Visual Studio Team System comprise were tested on thousands of user projects. Although your project may not be as big, you might still benefit from the team collaboration features.

This section walks you through some of the features of Visual Studio Team System, starting with managing a project, defining its architecture, and building and testing the software. The features of Visual Studio Team System are expanded on in Book VIII, Chapter 2.

Managing projects with Team System

Although the Team System doesn't include a special edition for project managers, it allows project managers to use Visual Studio to

- Create a project portfolio where you can access all the project's artifacts.
- Select a development methodology from predefined templates.
- Create a process-aware project portal complete with reports for evaluating progress.

A project manager views a project by using Portfolio Explorer in Visual Studio or the project's portal. By using Visual Studio Portfolio Explorer, you can

- Maintain source code control.
- Query work items, such as requirements, tasks, and bugs.
- View progress reports.
- View project documentation.
- Review build information.

Because the project portal is SharePoint-driven, you can open artifacts (such as requirements checklists) in Excel and modify them. Multiple people can then work on these artifacts and check out, publish, synchronize, and enforce data-validation rules.

Visual Studio Team System allows other team members to perform these tasks:

- **Create and edit work items, associate work items with source code, and view work item queues.** This task allows team members to manage work items and correlate those work items to source code.
- **Import work items into Microsoft Project and publish the updated project plan back to the project portal.** This task allows team members to put work items from Team System into a Microsoft Project file.
- **Sync updates from Project, Excel, and the portal to a team database.** This task allows developers to use Visual Studio to view and update work items.
- **Work with the tools and views that make them productive** (analysts in spreadsheets, developers in Visual Studio, and project managers in Project). Other stakeholders can use the project portal to monitor progress.
- **View reports on scheduling, bug fixes, and requirements churning.** Project managers can also drill into the reports to see underlying data.
- **Collect data throughout the course of a project as developers work.** You don't need to hold numerous status meetings and have people fill out and compile paperwork.

Architecting software

By using the designers in Visual Studio Team Edition for Software Architects, architects can

- Model applications and the target technology environment.
- Validate that the applications will run in the target environment.

These tools are geared primarily toward service-oriented architectures, such as Web services.

One incredible tool in Visual Studio Team Edition for Software Architects is Application Designer. You can use it to

- **Lay out applications that provide and consume new or existing Web services.**
- **Connect applications to build multi-tier applications.**
- **Show how to map applications to the logical datacenter design.**
- **Turn the diagram into code by creating projects and classes and stubbing out methods.**
- **Keep code synchronized with design in both directions.**

Application Designer is beneficial for these reasons:

- Architects can see the visual structure of an application.
- You can lay out the design without committing to code, also known as *whiteboarding*.
- You don't have to abandon design tools when you move to code.
- You can validate designs long before anything gets built.

The Software Architect Edition also provides a Logical Datacenter designer, which allows architects to map the applications they design to a technology environment that will host them. The Logical Datacenter allows architects to

- **Query their technology infrastructures.**
- **Define settings and constraints for the servers in their datacenters, such as what kind of authentication the server allows and whether the application needs scripting.**
- **Determine whether their applications will run in the target environment.**
- **Use graphical tools to change the configurations of applications or host environments.**
- **See communication pathways in the datacenter.**
- **Drag, drop, and validate by performing real-time validation to tell you which applications can be hosted on a given server.**
- **Specify settings and constraints in Application Designer for a component.**

The Application and Logical Datacenter designers open communications between designers, developers, and information technologists.

Developing software with Team System

The Software Developers Edition provides additional tools for testing and code analysis:

- **FxCop:** Enable code analysis and receive warnings as part of the build process.
- **Performance tools:** See which functions are called most often.
- **Unit testing and code coverage:** Test your code and use visual tools to see how much of your code is tested.

Testing with tools

The testing tools in Visual Studio Team Edition for Testers allows users to

- Record tests and generate scripts based on browser sessions.
- Play back scripts.
- Create validation rules to test for conditions on the Web pages.
- Bind data to tests so that you can use data-driven tests (for example, to search for keywords in a database).
- Use a load-test wizard that groups counters and offers guidance so that you're not guessing what each counter does and which one to use.

Collaborating with Visual Studio Team Foundation

The Team Foundation Server enables the collaboration features for the Team Editions of Visual Studio Team System. With TFS, you get these features:

- Source code control
- Work-item tracking
- Build automation
- Team communications
- Reporting

The integrated check-in procedure using TFS includes

- **Channels for managing source code, work items, check-in notes, and policy warnings.** *Check-in notes* are fields the project manager creates that the developer is required to fill out before check-in.
- **The option to create a standard check-in process.** You can review this process.
- **Access to the compare feature.** That way, you can expedite code reviews by comparing two different versions of source code.
- **Policy warnings notification.** For example, you're warned that a check-in must be associated with a work item.

Chapter 6: New to Visual Studio 2010

In This Chapter

- **Introducing parallel programming and cloud computing**
- **Increasing productivity for development teams**
- **Accessing more databases**

Visual Studio 2010 has been on the horizon for a long time. Even during the release of Visual Studio 2008, rumors of the next version (code named Hawaii) abounded. And now that Visual Studio 2010 is here, it's living up to expectations. This chapter talks about the new features, most of which are just about guaranteed to rock the world of Microsoft-platform developers.

Exploring Additional Programming Paradigms

There's always a lot of hype about technology, and most of it originates with the developers of the technology. The hype starts off with the developers, moves through managers, and then ends up with marketing. And it's the marketing spin that has delivered some of the most overblown technology promises.

I remember listening to news reports more than ten years ago about how object-oriented programming (OOP) was a major contributor to software profits. Don't get me wrong, I'm an OOP evangelist. But the effect of OOP on the bottom line was so exaggerated that it was laughable from my point of view. Yes, OOP saves companies money if managed correctly (and costs dearly, if not). But OOP isn't a reason for a stock to double or triple as some did in the 1990s.

In the following sections, I talk about two new paradigms that Visual Studio 2010 supports. Yes, there's hype out there about them, but these paradigms are truly worthy of mention. But if a stock doubles or triples because they're mentioned, sell the stock and take a profit.

Parallel programming

I teach programming and computer science. Early in the introductory courses, we talk about *algorithms.* These sequences of steps get something done. There's nothing magic about it. Baking a cake is an algorithm that follows a sequence.

One thing that is inherent to an algorithm is a single, monothreaded sequence of events. Most CPUs until recently only processed single simultaneous instructions. I remember getting a dual processor machine so that I could develop much faster. This was way before Core Duo and Core Quad processors. In order to have two processing centers, you needed to add an additional CPU.

Multicore processors are essential for modern software applications that often perform complex algorithms, access databases across a network, and update a graphical interface simultaneously.

In years past, if an application needed to perform several simultaneous tasks, the programmers spun up a separate threads that each managed a task. The operating system then managed switching CPU attention between all the threads. For software that did things such as listen on a socket, multiple threads were absolutely essential. A well-known example is Microsoft Word's spell checker. You may have noticed the Word doesn't always underline misspelled words in documents the instant that you open them. The reason for the delay is that Word spins up a separate thread to do the spell checking, and it sometimes takes awhile to complete the check for a long document.

Visual Studio 2010 has gone way beyond giving you the ability to create multiple threads. It provides a way to assign tasks to specific CPUs or Cores. In this way, you can manage your application's performance in a much more controlled way rather than rely on the operating system to switch between the threads.

Visual Studio 2010 and the .NET 4.0 framework introduce two new technologies that allow the programmer to optimize their managed code for multicore processors. The Task Parallel Library (TPL) is designed to make it easier to write managed code that automatically use multiple processors. With this library, you can easily express potential parallelism in existing sequential code where the parallel tasks will execute concurrently on available processors. The Parallel Language Integrated Query (PLINQ) is a query execution engine that accepts any LINQ-to-Objects or LINQ-to-XML query and executes them on available processors.

For more information and examples on how to use these components, read Book V, Chapter 6.

Cloud computing

I work on a lot of simultaneous projects. The documents that support these projects are design documents, spreadsheets with hours for billing, invoices, and much more. I travel a lot and don't always have access to the document repository. For that reason, my documents tend to be in several places, which can mean serious trouble for me when I can't find an important document that I need immediately.

For all these reasons, I jumped up and down when Google created their Google Documents. Now I keep all my project documents within the Google Documents space. No matter where I am, as long as I have an Internet connection, I have my documents.

My documents reside in a space somewhere in the Internet. Sometimes you see the Internet represented as a cloud in diagrams. It's a good metaphor. And the cloud metaphor has been extended to things, such as Google Documents. The space Google Documents and similar applications occupy is known as *cloud computing.*

I recently got a call from a company for which I have consulted for many years. The person explained that the company's application relied on the cloud and that he needed me to write some ancillary cloud computing applications.

Although I have written cloud computing applications in the past, it's not that easy. You never know the physical space in which a file resides. It may even be split among physical servers. All these issues can really be challenging. So when I got the call, I had some initial apprehension.

The latest release of Visual Studio addresses the complexity of cloud computing applications. It comes complete with classes that seamlessly deal with cloud computing applications. And it also comes with a local simulator so that you can develop applications without having to connect to an actual cloud during development. It all relies on a technology named Azure. You can find much more on this topic in Book V, Chapter 5. You'll be amazed at how easily you can develop applications for the cloud.

Reaping the Benefits for Development Teams

Visual Studio is a tool that makes developers much more productive and offers features such as IntelliSense and an integrated debugger. Visual Studio 2010 enhancements offer fantastic productivity to developers. The following sections underscore the best of the new features.

Enhanced user interface

A better user interface is important, especially to newcomers to Visual Studio, because the pre-2010 versions all had incremental user interface changes. Visual Studio veterans will take some time to get used to the new features. It reminds me of when I first sat down with Microsoft Office 2007. I hated it for the first 30 minutes because it was so different than Office 2003. But after the initial period, I started loving it because it was so much better. Visual Studio 2010's new user interface will be the same, and after a short time, you will love it.

The entire user interface is much less cluttered. Things are organized in well-thought-out ways. You can move and position windows in more flexible ways, thus meeting more needs than the previous versions.

The editor is improved, too. It has enhanced real-time syntax checking. IntelliSense has been beefed up, especially for Visual C++.

Better development lifecycle management

Many modern software applications aren't created by a single developer, but are developed by large, often distributed, teams of developers, testers, product managers, and other managers that all must collaborate to produce a high-quality product that gets to market in a timely fashion. How do you ensure that code is properly managed among all developers, defect reports are submitted by testers in the QA process and addressed by developers, and marketing requirements are provided to managers and developers? In general, the entire process is known as *Application Lifecycle Management* (ALM), and Visual Studio has stepped in with powerful tools.

One of the versions of Visual Studio 2010 is Visual Studio Team System 2010. It handles all aspects of the application lifecycle, including development, testing, and enhancements.

One of the biggest problems that I have experienced in the cycle happens during testing. I have literally had testers knock at my door and say, "It doesn't work." Then I have to ask a list of questions (kind of like the 20 questions game) and find out what is really going on. Testers aren't always good communicators, which in turn means that developers don't get the problems fixed as well or as quickly as they can. Another difficulty is that the application may work on the developer's system and not on the tester's system.

A tool called TestRunner is part of the Team System and can record everything necessary for the developer to recreate the problem. In other words, the testers go through the QA process and create data almost like a play-by-play recording. They developers can use this data to pinpoint the issues that arose.

TestRunner is actually part of an overall system for this historical debugging known as Camano. This set of features is sure to reduce the development time and increase the robustness of released software.

Better programming productivity

The more efficient that an IDE is, the faster the project gets developed, saving serious resources for any company. Visual Studio 2010 has gone way beyond expectations to give you something that makes the development experience better and faster.

IntelliSense, the feature that offers completion of the token that you're currently typing, is better. It's faster, more complete, and much better for Visual C++. It picks up classes, methods, properties, and variables that are part of your code, not just the .NET framework.

Have you ever programmed in C# or C++ and wondered where the matching brace was? With complicated code, it's sometimes hard to match up the curly braces. But Visual Studio 2010 will show you exactly where a matching brace is, saving you hours of analysis as you try to get your code right.

Better Web development tools

Just the introduction of ASP.NET has really improved Web development. But Visual Studio 2010 has added features that make developing Web applications even easier. For starters, a one-click deployment feature sends all updated files to your Web server. This feature ends up saving me a lot of time because I don't have to run an FTP client and upload the files, making sure only to send the ones that have been updated.

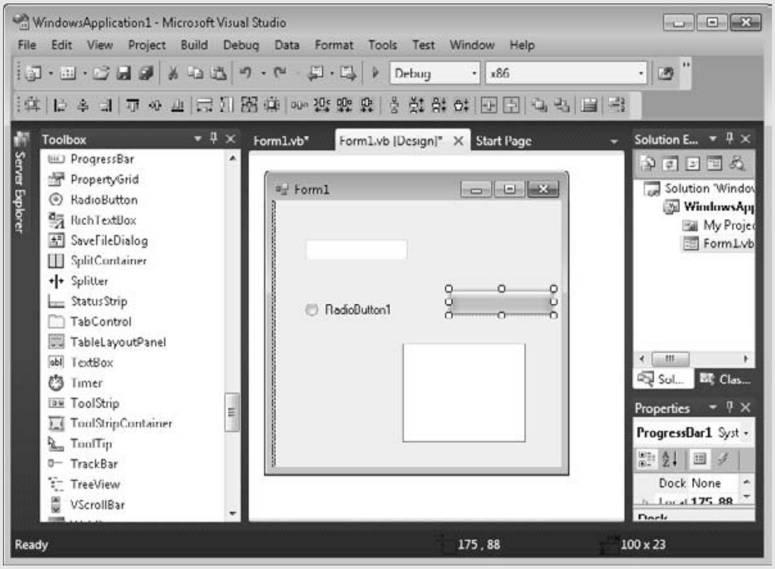
Silverlight, Microsoft's rich content type, is easy to develop for. It integrates seamlessly into applications. I would bet that with Microsoft behind it, Silverlight will start giving Flash content a run for its money.

Gaining Databases Support

Accessing a SQL Server database from .NET applications has always been easy. And going one step further, accessing any other database via an ODBC connection was also somewhat easy. But this extra layer added complexity and some dependencies and took away performance. Now Visual Studio provides native clients for IBM's DB2 database and Oracle databases. Adding the support for these two popular databases has opened the door for .NET applications to be developed for those two databases, where before accessing these databases wasn't practical without the support.

Book II

Getting Started with Visual Studio



Contents at a Glance

Chapter 1: Installing Visual Studio Professional

In This Chapter

- **Using multiple versions of Visual Studio**
- **Meeting the system requirements for Visual Studio**
- **Installing Visual Studio**
- **Launching Visual Studio**

As long as you have adequate hardware resources to run Visual Studio, getting it up and running is a straightforward process. You might spend all day waiting for the software to install, but then you can stay up all night playing around with all its cool new features. This chapter gives you the details.

Installing Versions Side by Side

You can use any release version of Visual Studio, from Visual Studio 6.0 through Visual Studio 2010, on a single computer. Of course, I don't recommend opening all the versions at one time. Here are the versions:

- Visual Studio 6.0
- Visual Studio .NET
- Visual Studio .NET 2003
- Visual Studio 2005
- Visual Studio 2008
- Visual Studio 2010

If you installed beta versions or any other prerelease version of Visual Studio 2010 on your computer, you should uninstall them completely before installing the final version of Visual Studio 2010.

WARNING!

As a rule, after you open an older version of a solution in the next version of Visual Studio, you can no longer open the solution in the previous version. For example, if you open a Visual Studio 2005 solution in Visual Studio 2010, you can no longer open the solution in Visual Studio 2005. Visual Studio makes a backup of the files it changes as part of the upgrade process (unless you tell it not to do so). Nevertheless, it's a good idea to have backups of all your solutions before you start opening them in different versions.

Taking a look at all the .NETs

You can have multiple versions of the Microsoft .NET Framework installed on a single computer. These versions of .NET have been released:

- Version 1.0, released with Visual Studio .NET
- Version 1.1, released with Visual Studio 2003
- Version 2.0, released with Visual Studio 2005
- Version 3.0, released with Windows Vista, Windows 2008, and products such as System Center Operations Manager (SCOM)
- Version 3.5, released with Visual Studio 2008
- Version 4.0, released with Visual Studio 2010

Visual Studio 2010 allows you to target your development to any version of the .NET framework.

Book II, Chapter 4 talks about targeting multiple versions of the .NET Framework with Visual Studio 2010.

Getting help from Visual Studio 2010

Visual Studio 2010 introduces the Help Library Manager, which is a separate application from Visual Studio. The Help Library Manager features a local help database and integrates with online resources.

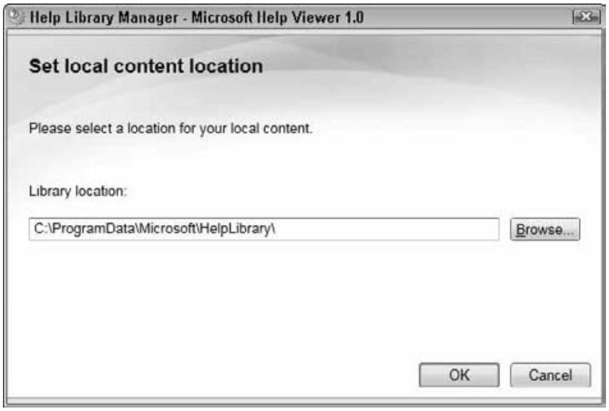
To use the Help Library Manager in Visual Studio 2010, follow these steps:

1. **Choose Help➪Manage Help Settings.**

   The Visual Studio 2010 Help Library Manager dialog box appears, as shown in Figure 1-1. The Help Library manager prompts you to choose a location for offline help storage if you haven't already specified it.

2. **Choose your location by clicking the Browse button to navigate to a folder.**

   Alternatively, you can type the location in the edit box.



Help Library Manager - Microsoft Help Viewer 1.0

Set local content location

Please select a location for your local content.

Library location:

C:\ProgramData\Microsoft\HelpLibrary\

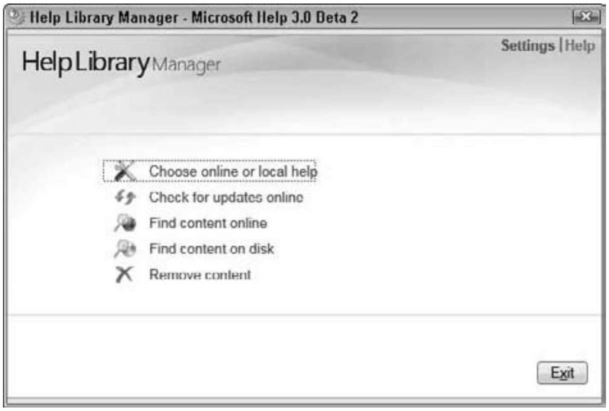
Figure 1-1: Configure local help content location.

3. **Click OK.**

   The Set Local Content Location dialog box closes. You see the Help Library Manager dialog box, shown in Figure 1-2.

   WARNING!

   After you choose a location for offline help, you can't change it, so avoid using removable media or folders used by Windows or other applications.

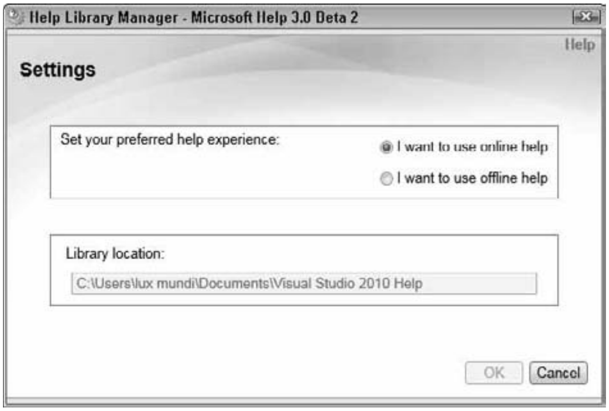


Help Library Manager - Microsoft Help 3.0 Deta 2

Figure 1-2: The Visual Studio 2010 Help Library Manager.

4. **Select Choose Online or Local Help.**

   The Help Library Manager Settings dialog box shown in Figure 1-3, appears.



Set your preferred help experience:

● I want to use online help

○ I want to use offline help

Library location:

C:\Users\lux mundi\Documents\Visual Studio 2010 Help

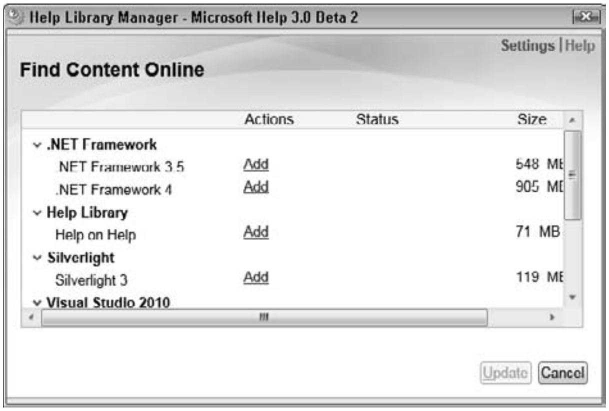
Figure 1-3: The Help Settings dialog box.

5. **Choose the I Want to Use Online Help radio button and click OK.**

   The Help Settings dialog closes, and you return to the Help Library Manager dialog box (refer to Figure 1-2).

6. **Select Find Content Online.**

   The Find Content Online dialog box, shown in Figure 1-4, appears.



	Actions	Status	Size
.NET Framework			
.NET Framework 3.5	Add		548 MB
.NET Framework 4	Add		905 MI
Help Library			
Help on Help	Add		71 MB
Silverlight			
Silverlight 3	Add		119 MB
Visual Studio 2010			

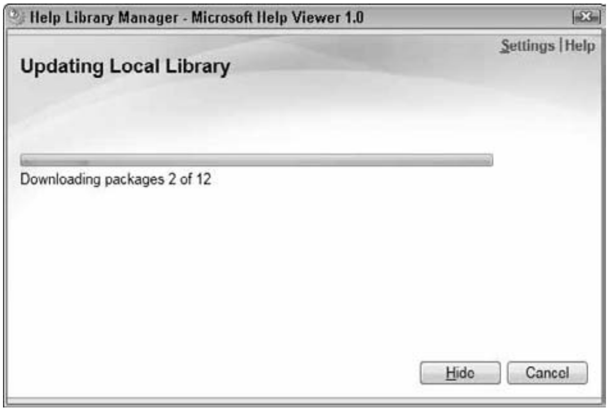
Figure 1-4: The Help Library Manager Find Content Online dialog box allows you to add online content to your help system.

7. **Choose the online resources you want for your help system by clicking the Add link that corresponds to the resource you want.**

   The Update button becomes enabled at the bottom of this dialog box.

8. **Click the Update button.**

   Visual Studio downloads the requested content, as shown in Figure 1-5.


Help Library Manager - Microsoft Help Viewer 1.0

Updating Local Library

Downloading packages 2 of 12

Figure 1-5: Visual Studio downloads the requested help library content.

9. **Repeat Steps 7 and 8 for the content you want to download.**
10. **When finished, click Exit on the Help Library Manager dialog box.**

Meeting System Requirements

Here are the minimum requirements for installing Visual Studio 2010 Professional Edition:

- A 2.0 GHz Core Duo processor
- 3GB RAM
- 7.5GB disk space
- A DVD-ROM drive

You also need one of these operating systems:

- Windows Server 2003
- Windows Server 2008
- Windows XP
- Windows Vista
- Windows 7

As usual, the minimum requirements probably aren't where you want to be. Although Visual Studio will surely run on a computer using the minimum requirements (and actually less than that), you'll probably be able to drink a cup of coffee between mouse clicks, which defeats the purpose of using productivity software. In reality, you probably want at least a 3.0 GHz Duo Core processor with at least 3GB RAM. If you don't think that you have enough hardware and can't upgrade, consider using the Express editions of Visual Studio.

If you're stuck running Visual Studio 2010 on an older machine that doesn't meet the requirements, you can add more memory to the system. I am currently running Visual Studio 2010, SQL Server 2008 Developer Edition, and all the other software you see in this book on a machine with 3.0 GHz dual-core processor and 4GB of memory.

You must have Administrator rights to the computer on which you want to install Visual Studio 2010. User rights are sufficient to run Visual Studio 2010.

Stepping through the Installation

Installing Visual Studio is fairly simple. After you insert the DVD, you see the Visual Studio 2010 Setup screen.

Installing Visual Studio is more involved than clicking `setup.exe` and walking through a wizard. Visual Studio requires a number of components to install in order to be fully functional. Installation involves the following tasks:

1. **Install Visual Studio and version 4.0 of the .NET Framework.**
2. **Install the help documentation.**
3. **Check for updates at Windows Update.**

To start the Installation Wizard and install Visual Studio 2010, follow these steps:

1. **Click the Install Visual Studio 2010 option on the Visual Studio 2010 Setup screen.**

   Setup loads the installation components and displays the Welcome screen.

2. **After the setup components are loaded, click Next on the Welcome screen.**

   The Microsoft Visual Studio 2010 Setup start page is displayed.

   The start page displays a list of components that are already installed and a list of components to be installed.

3. **On the start page, select the I Accept the Terms of the License Agreement check box to indicate that you read and agree to the license agreement.**

4. **Click Next.**

   The options page displays.

5. **On the Options page, select the radio button indicating whether you want the Default, Full, or Custom installation and then specify the installation path.**

   I almost always do a Full installation.

   In the lower-right corner, note the table that displays disk space requirements. This figure shows installing Visual Studio on the C: drive. You can, however, install to any active hard drive on a computer that has sufficient hard drive space. Visual Studio installs components on your system drive, even if you choose to install Visual Studio on another hard drive.

6. **Click Install on the Options page to start the installation.**

   The install page displays.

   If you chose the Custom installation from the Options page, you see the Next button rather than Install. After clicking Next, you can select all the individual components you want to install.

   The Install page displays the progress of the installation along with useful information about new features in Visual Studio 2010. After all the components are installed, the Finish page is displayed.

   If everything installs properly, the Finish page displays a message stating that setup is complete. If any errors occur, they're displayed on this page.

7. **Click the Finish button to close the Installation Wizard.**

   After the wizard closes, the Visual Studio 2010 Setup window displays again.

8. **You can click the link to install the product documentation.**

   After the documentation is installed, the Setup window is displayed again.

9. **Click the link to check for service releases and install any service release that Windows Update displays for you.**

Launching Visual Studio for the First Time

When you launch Visual Studio for the first time, you see the Choose Default Environment Settings dialog box. You can choose a set of environment settings based on the kind of development work you do.

Your choices for development settings are

- General
- Visual Basic

- Visual C#
- Visual C++
- Visual F#
- Web

The settings you choose determine the headings you see on the start page and the types of project templates you see when you create a new project and set your default programming language. The General option is a good choice because you might develop applications in more than one language. Development settings also let you customize menus, toolbars, and help documentation to suit the choice you make.

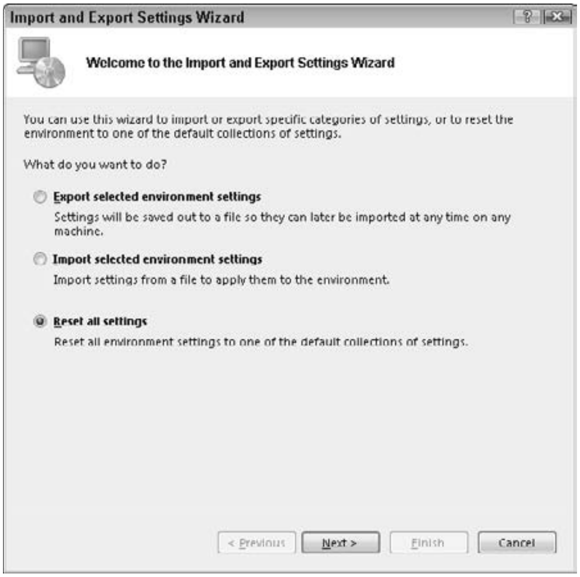
TIP

This book uses General Development Settings for all examples. In most cases, the keyboard shortcut commands are offered, when available. The keyboard shortcuts are pretty consistent across the development settings.

You can use the Import and Export Settings Wizard to change your development settings. To start the wizard, follow these steps:

1. **Choose Tools⇨Import and Export Settings.**

   The Import and Export Settings Wizard, shown in Figure 1-6, appears.



Import and Export Settings Wizard

Welcome to the Import and Export Settings Wizard

You can use this wizard to import or export specific categories of settings, or to reset the environment to one of the default collections of settings.

What do you want to do?

○ **Export selected environment settings**
Settings will be saved out to a file so they can later be imported at any time on any machine.

○ **Import selected environment settings**
Import settings from a file to apply them to the environment.

● **Reset all settings**
Reset all environment settings to one of the default collections of settings.

Figure 1-6: Use the Import and Export Settings Wizard to change your default settings.

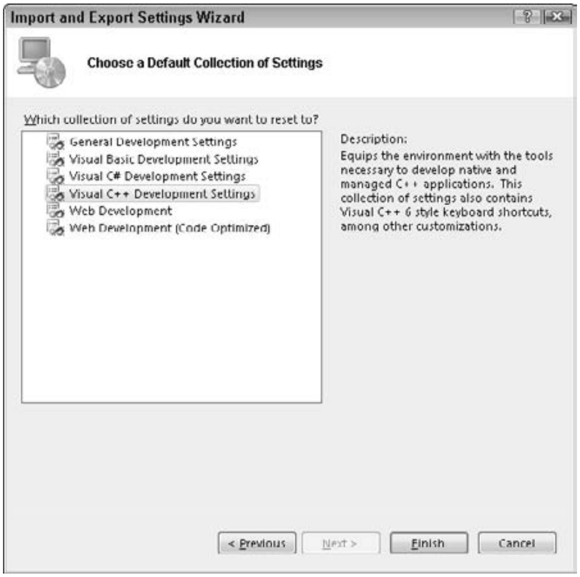
2. **Select the Reset All Settings radio button and then click Next.**

   The Save Current Settings page is displayed.

3. **To save your settings so that you can reuse them later, click Yes and then specify the filename and directory; to discard your settings, click No.**

4. **Click Next to save or discard your settings.**

5. **On the next page that displays, choose a default collection of settings from the list there, as shown in Figure 1-7.**



Import and Export Settings Wizard

Choose a Default Collection of Settings

Which collection of settings do you want to reset to?

- General Development Settings
- Visual Basic Development Settings
- Visual C# Development Settings
- Visual C++ Development Settings
- Web Development
- Web Development (Code Optimized)

Description:
Equips the environment with the tools necessary to develop native and managed C++ applications. This collection of settings also contains Visual C++ 6 style keyboard shortcuts, among other customizations.

Figure 1-7: Select a collection of settings to reset as your default settings.

6. **Click Finish.**

   Visual Studio changes your settings.

TIP

You can also use the Import and Export Settings Wizard to import settings that you saved from another machine or that a friend sent you.

Chapter 2: Browsing Visual Studio

In This Chapter

- **Becoming familiar with the development environment**
- **Understanding solutions and projects**
- **Browsing properties, servers, and code**
- **Painting forms with visual designers**

With Visual Studio 2010, you have access to hundreds of commands, tools, and designers. The Visual Studio Code Editor and Object Browser put the Microsoft .NET Framework's thousands of methods at your fingertips. You have everything you need to build powerful, world-class Windows applications and Web sites — if only you knew where to begin.

Making Your Way around Visual Studio

When you first open Visual Studio 2010, you see a screen that reflects a Visual Studio 2010 installation that uses the general development settings.

Note these features

- **Start page:** This page is your dashboard for navigating Visual Studio 2010 and connecting to resources in the .NET community. The start page includes these features:
  - *Recent Projects:* A list of recently opened projects in Visual Studio 2010; also includes links to open and create new projects and Web sites
  - *Getting Started:* A list of links to resources to help you get started using Visual Studio 2010
  - *Guidance and Resources:* A list of links relevant to Visual Studio 2010 development processes, MSDN resources, and additional tools
  - *Latest News:* An RSS feed of the latest MSDN news
- **Solution Explorer:** This task pane on the right side of the screen lists open solutions and the projects they contain.
- **Class View:** This tab on the task pane in the lower-right corner of the screen displays a tree view of classes contained in the open solution.

- **Server Explorer:** This task pane on the left side of the screen provides a tree view of the servers you're working with. The Server Explorer is automatically hidden by default, but it appears if you hover your mouse over it.
- **Toolbox:** This task pane on the left side of the screen provides groups of controls that you can drag and drop onto visual designers in Visual Studio 2010.
- **Menu bar and Standard toolbar:** These two elements provide access to additional tools and windows.

Because most of the task panes, windows, and menus in Visual Studio 2010 are context sensitive, they display different options depending on what kind of application you're developing. For example, the toolbox displays text boxes and buttons that work in Windows applications when you open them.

Visual Studio provides many windows in addition to the ones displayed by default. You can display all the windows by using the View menu, as shown in Figure 2-1.

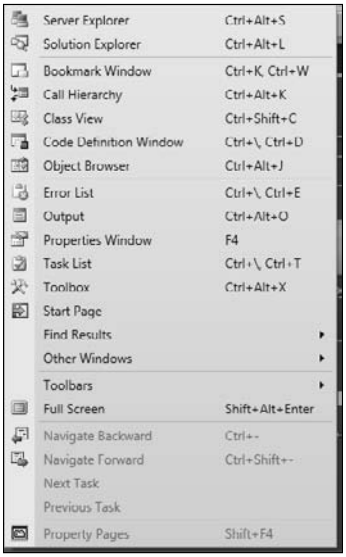

Server Explorer Ctrl+Alt+S
Solution Explorer Ctrl+Alt+L
Bookmark Window Ctrl+K, Ctrl+W
Call Hierarchy Ctrl+Alt+K
Class View Ctrl+Shift+C
Code Definition Window Ctrl+\, Ctrl+D
Object Browser Ctrl+Alt+J
Error List Ctrl+\, Ctrl+E
Output Ctrl+Alt+O
Properties Window F4
Task List Ctrl+\, Ctrl+T
Toolbox Ctrl+Alt+X
Start Page
Find Results
Other Windows
Toolbars
Full Screen Shift+Alt+Enter
Navigate Backward Ctrl+-
Navigate Forward Ctrl+Shift+-
Next Task
Previous Task
Property Pages Shift+F4

Figure 2-1: Use the View menu to open additional windows.

Dealing with Windows Galore

Visual Studio 2010 displays content in two basic Window styles:

- Tabbed documents, such as the start page
- Task panes, such as Solution Explorer

You can treat anything that opens as a tabbed document like a task pane and vice versa. Here are your options for working with windows in Visual Studio 2010:

- **Floating:** The window floats within the environment.
- **Dockable:** You can dock the window in a certain area of the screen.
- **Tabbed Document:** The window is placed in the tabbed-documents area.
- **Auto Hide:** You can toggle the window between minimized and restored.
- **Hide:** The window is hidden from view.

To access the window options, follow these steps:

1. **Right-click the title bar of the window you want to manipulate.**
2. **Choose the window option from the contextual menu.**

TIP

You can also access the window options by using the Window menu.

If you change your mind or make a mistake and want to restore all the windows to their original states, follow these steps:

1. **Choose Window➪Reset Window Layout.**

   A confirmation box appears and asks you to confirm that you want the windows reset to the default layout.

2. **Click Yes.**

   All windows are restored.

Docking windows

When you make a window dockable, you can use positioning anchors to position it on-screen. To dock a tabbed document, follow these steps:

1. **Right-click the tabbed document and choose Dockable from the contextual menu.**

   The window appears as a floating window.

   If the Dockable option is already selected (a check mark appears next to it), the window is ready to be docked. Go on to Step 2.

2. **Position your mouse over the title bar of the window and start dragging the window.**

   While you drag, position anchors appear on-screen.

3. **Move the window toward a position anchor.**

   As you hover over a position anchor, a shadow is placed where the anchor will position the window.

4. **Release the window over the position anchor.**

   The window is docked.

Working with tabbed documents

You can display any window as a tabbed document by simply right-clicking the window's title bar and choosing Tabbed Document from the contextual menu. When more than one document is displayed, you also have the options of creating horizontal and vertical groups of tabbed documents. To create a vertical tab group, follow these steps:

1. **Right-click the tab of a tabbed document.**
2. **Choose New Vertical Tab Group from the contextual menu that appears.**

   The documents are displayed side by side in the viewing area.

You can create horizontal groups by choosing New Horizontal Tab Group. To remove the groupings, right-click the tab of the tabbed document and choose Move to Previous Tab Group from the contextual menu.

Managing windows

You can always use the Window menu to manage your open windows. Figure 2-2 shows the list of commands available on the Window menu.

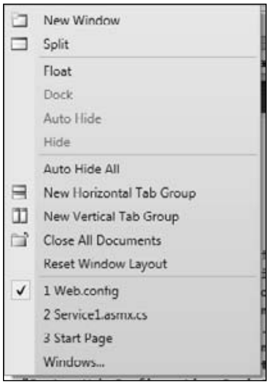


Figure 2-2: Use the Window menu to manage your open windows.

Building Solutions with Visual Studio 2010

Exploring Visual Studio without any files open is different from exploring it when you have an application open for editing. Most windows and task panes aren't populated with options until files are open. Although you can use Visual Studio to open individual files (such as XML files), in most cases, you use Visual Studio to create new Windows and Web applications and edit existing applications.

Applications you create with Visual Studio 2010 require many different kinds of files in order to work properly. The files that make up an application are items. Examples of items include

- Source code files
- References
- XML and HTML files
- Visual designer settings
- Data files

The items that make up an application are grouped into containers so that the items are easier to manage. Visual Studio 2010 provides two types of containers:

- **Project:** A group of items that combine to create a component of a solution, such as a Windows project
- **Solution:** A collection of projects and items that combine to create a deployable solution

You use the New Project dialog box, as shown in Figure 2-3, to create new projects and solutions.

You can invoke the New Project dialog box by choosing File➪New➪Project; clicking a link on the start page; or from a contextual menu in the Solution Explorer.

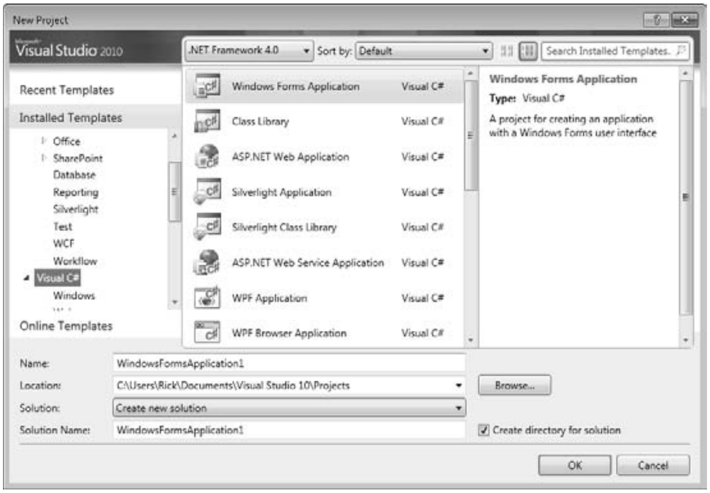
Exploring the New Project dialog box gives you some idea of the kinds of applications you can create using Visual Studio 2010. To open the New Project dialog box by using the Recent Projects area, follow these steps:

1. **Click the start page.**

   If it isn't displayed, choose View➪Start Page.

2. **In the New Project area of the start page, click the Installed Templates link.**

   The project types for that template set displays in another window.


New Project
Visual Studio 2010
Recent Templates
Installed Templates
Office
SharePoint
Database
Reporting
Silverlight
Test
WCF
Workflow
Visual C#
Windows
Online Templates
Windows Forms Application Visual C#
Class Library Visual C#
ASP.NET Web Application Visual C#
Silverlight Application Visual C#
Silverlight Class Library Visual C#
ASP.NET Web Service Application Visual C#
WPF Application Visual C#
WPF Browser Application Visual C#
Search Installed Templates
Browse...
OK
Cancel

.NET Framework 4.0

Sort by: Default

Windows Forms Application

Type: Visual C#

A project for creating an application with a Windows Forms user interface

Name: WindowsFormsApplication1

Location: C:\Users\Rick\Documents\Visual Studio 10\Projects

Solution: Create new solution

Solution Name: WindowsFormsApplication1

☑ Create directory for solution

Figure 2-3: Create new projects and solutions from the New Project dialog box.

The New Project dialog box (refer to Figure 2-3) has three areas:

- **Project templates:** Displays the programming languages and kinds of projects you can create for each language. This area is in the left pane of the New Project dialog box.
- **Project types:** Displays the list of project templates available for each type of project. This is in the middle pane of the New Project dialog box.

  Visual Studio uses project templates to know what files to create when you tell it to create a certain kind of application. For example, when you use the New Project dialog box to create a new Windows application, the Windows Application project template tells Visual Studio to create a Windows project with a new, blank Windows Form.
- **Project and solution details:** Allows you to name the project and tell Visual Studio where you want it saved and allows you to specify whether you want a solution created and what to name it. By default, Visual Studio always creates a solution to contain your projects.

Here's how to create a new project by using the New Project window:

1. **In the Installed Templates tree, expand the Visual C# project type category.**

   A list appears of project types that you can create by using the Visual C# language.
2. **Click the Windows Forms Application project type.**

   A list of project templates that are available for Windows Forms applications appears in the Templates window.

3. **Click the Windows Forms Application template.**
4. **Enter a filename for your new Windows Forms application in the Name text box.**
5. **Accept the default file location or click the Browse button to select a new location.**
6. **Click OK to create the new project.**

   A new Visual C# project is created, and a blank Windows Form is displayed in the Forms Designer.

Using the Solution Explorer to manage solutions and projects

The Solution Explorer graphical tool helps you manage solutions and projects. When you open a solution or project, a graphical representation of the solution and the items within it are displayed in the Solution Explorer. In Figure 2-4, Solution Explorer displays the new Windows application created in the preceding section.

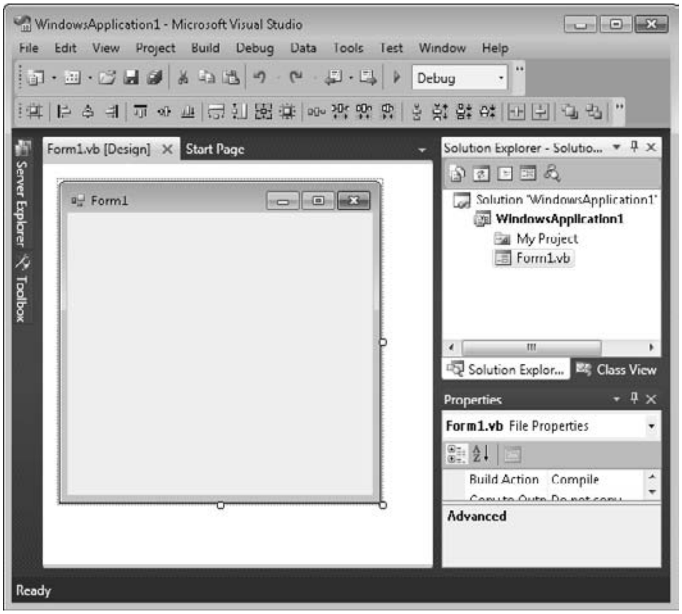


Figure 2-4: Use Solution Explorer to manage an open solution and its projects.

Solution Explorer provides many commands you can use to manage your solution. To view a list of commands available for managing a solution, follow these steps:

1. **Right-click the solution in the Solution Explorer.**

   The solution is the topmost item in the tree. A contextual menu appears.

2. **Choose a command from the menu.**

You can also right-click a project in Solution Explorer to view a contextual menu of project commands. Figure 2-5 shows the contextual menus for solutions (left) and projects (right).

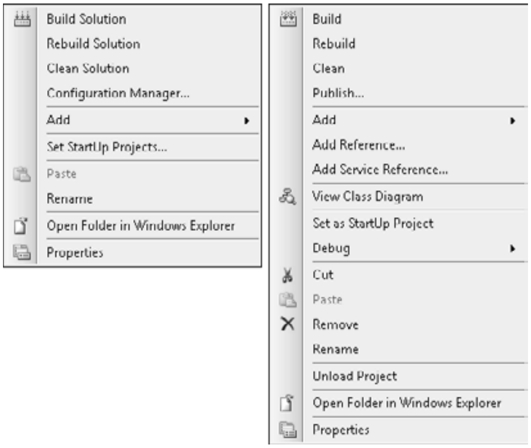


Figure 2-5: Right-click a solution (left) or project (right) to display its contextual menu.

On each of the contextual menus, the Add command expands to display a list of items you can add to your solution or project. Items you can add to a solution include

- ✦ New and existing projects
- ✦ New and existing Web sites
- ✦ New and existing items, such as XML files
- ✦ Folders for organizing items
- ✦ Items you can add to a Windows project include
- ✦ New and existing items, such as Windows Forms, classes, and XML files
- ✦ Folders for organizing items

The items that you can add to other kinds of projects are specific to the type of project. For example, you can add Web Forms to Web sites.

You can right-click the items within a project to display contextual menus, too. For example, when you right-click a Windows Form, you see a menu that displays the following commands to

- View Code
- View Designer
- View Class Diagram

Working with solution and project menus

When you open a solution in Visual Studio 2010, the menu bar displays project-specific menus, including these three:

- **Project or Website:** Manage Windows projects or Web sites.
- **Build:** Build and deploy your solution.
- **Debug:** Use the Debugger.

Depending on the kind of project you open, you might also see menus for Data, Format, and Layout. When you're using Visual Studio 2010, don't forget to look up at the menu bar occasionally to see additional commands that you may not have known exist.

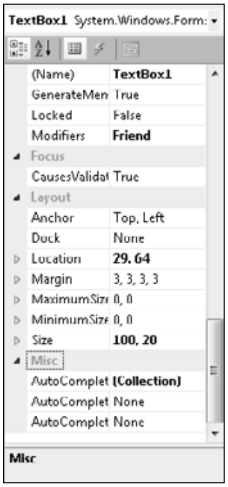
Using the Properties Window

Almost everything in Visual Studio — from controls to solutions — has properties. Properties can be quite simple, such as a few settings options, or they can be composed of complex configuration wizards. The Properties window is displayed by default as a task pane on the right side of the screen.

You can view the properties of an item in several ways:

- Right-click an item and choose Properties from the contextual menu.
- Click an item and choose View➪Properties Window.
- Click an item and press F4.

Figure 2-6 shows the Properties window for a text box control for a Windows Forms Application. To read more about using properties, see Book III, about building applications with Visual Studio 2010.



TextBox1 System.Windows.Form:

(Name)	**TextBox1**
GenerateMen	True
Locked	False
Modifiers	**Friend**
◢ Focus	
CausesValidat	True
◢ Layout	
Anchor	Top, Left
Dock	None
▷ Location	**29, 64**
▷ Margin	3, 3, 3, 3
▷ MaximumSiz	0, 0
▷ MinimumSiz	0, 0
▷ Size	**100, 20**
◢ Misc	
AutoComplet	**(Collection)**
AutoComplet	None
AutoComplet	None

Misc

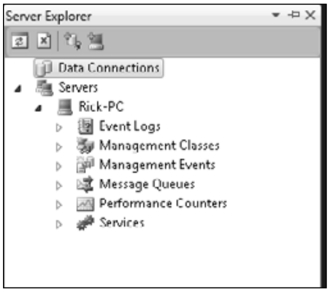
Figure 2-6: Set the properties for an item via the Properties window.

Browsing Servers

Use Server Explorer to view server configuration details and to connect to data sources. Server Explorer is displayed by default as a task pane on the left side of the screen. You can use Server Explorer for these tasks:

- Create and manage data connections.
- Drag and drop items, such as event logs and message queues, from a server to the design surface.
- View configuration details about services running on a server.

Figure 2-7 shows an example of Server Explorer. (See Book IV, Chapter 3 for more information about using Server Explorer to manage data connections.)



Server Explorer

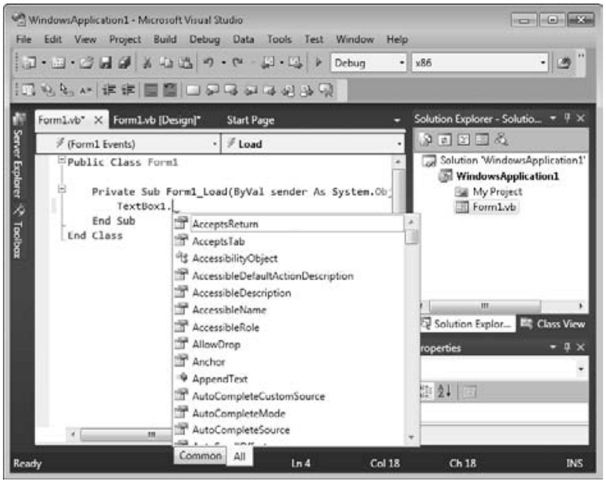
- Data Connections
- ◢ Servers
  - ◢ Rick-PC
    - ▷ Event Logs
    - ▷ Management Classes
    - ▷ Management Events
    - ▷ Message Queues
    - ▷ Performance Counters
    - ▷ Services

Figure 2-7: Use Server Explorer to manage data connections and other server resources.

Writing Code with the Code Editor

Building applications requires writing code. For this purpose, Visual Studio 2010 provides a code editor, which is a special kind of word-processing program. Unlike the word processor used to write this book, a code editor is tuned to write programs. Although a regular word processor includes features like table formats and bullet points, the Visual Studio 2010 Code Editor has these features:

- **Auto-complete:** The Code Editor provides an intelligent authoring and editing auto-complete tool named IntelliSense. It provides you with a list of programming commands that are appropriate for the context in which you're writing. Figure 2-8 shows the Code Editor with IntelliSense in action.



```
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Ob
        TextBox1.
    End Sub
End Class
```

```
AcceptsReturn
AcceptsTab
AccessibilityObject
AccessibleDefaultActionDescription
AccessibleDescription
AccessibleName
AccessibleRole
AllowDrop
Anchor
AppendText
AutoCompleteCustomSource
AutoCompleteMode
AutoCompleteSource
```

Figure 2-8: The Code Editor is a special word processor for writing computer programs.

- **Formatting:** Write consistent-looking code using features to indent, wrap words, and place tabbed blocks of code.
- **Outlining:** Collapse blocks of code so that they're out of view.
- **Macros:** Use Visual Studio 2010's helper tools, such as code snippets and comment/uncomment features.

Visual Studio 2010 provides intelligent code editors for

- .NET programming languages, such as Visual Basic, C#, and F
- Web-based files, such as HTML, CSS, and XML
- General-purpose languages, such as C++

To open the Code Editor, use one of these methods:

- Open a file that uses any of the programming languages just mentioned.
- Double-click a code file in Solution Explorer.
- Add a new programming file to a solution.

Book V shows you the Code Editor in action.

Using the Forms Designer

One key feature of Visual Studio 2010 is that you can build a Windows application or Web site by dragging and dropping controls, such as text boxes and labels, onto a design surface. The designer generates, behind the scenes, the code that's necessary to create the user interface you're painting with the designer. Visual Studio 2010 provides three distinct forms designers:

- Windows Forms Designer
- Web Forms Designer
- Windows Presentation Foundation (WPF) Designer

The forms designers are displayed by default when you create a new Windows application or Web site.

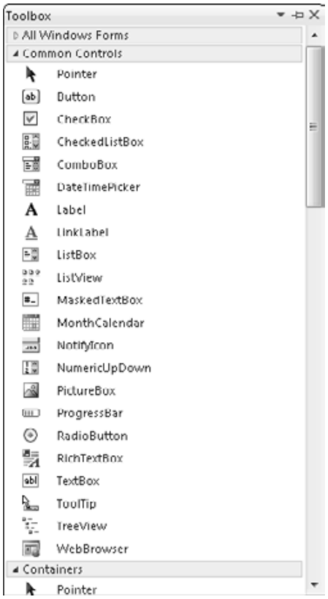
Two windows that go hand in hand with the forms designers are the toolbox and the Properties window. (See the section "Using the Properties Window," earlier in this chapter.) The toolbox, displayed by default as a task pane on the left side of the screen, provides a list of the controls you can drag and drop onto the forms designer to create a Windows Form, Web Form, or WPF Form. Figure 2-9 shows the Windows Forms Designer and the toolbox.

See Book III for examples of the visual forms designers.

Taking a Look at Other Designers

Visual Studio provides many visual designers that are intended to increase your productivity. Here are a few:

- **Class Designer:** Model existing code or create new code by building models. Book V, Chapter 3 describes the Class Designer.
- **DataSet Designer:** Drag and drop tables from a database to create programmable datasets. See Book IV, Chapter 3.



Toolbox
All Windows Forms
Common Controls
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
TreeView
WebBrowser
Containers
Pointer

Figure 2-9: Use the toolbox to drag controls and drop them on the forms designer.

- **Crystal Reports:** Visually build reports for Windows or Web applications. Book VI, Chapter 4 demonstrates how to use the Crystal Reports Designer.
- **Smart devices:** Create applications for smartphones and use emulators to see what the software does. Start building applications for smart devices in Book III.

Chapter 3: Making Connections with Visual Studio

In This Chapter

- **Exploring the help features of Visual Studio 2010**
- **Reaching out to the Windows developer community**
- **Keeping up with certifications and Webcasts**

When you use Visual Studio, you're not just using a development tool — you're joining a community. This statement is true, no matter what development tool you use. The technologies people use often cluster them into communities, such as Java, Linux and open source software, Mac, and .NET and other Windows developers.

That's not to say that people don't cross over — because they do. Many Windows developers are also closet Linux folks. Nevertheless, as a Windows developer, you're likely to run in the same circles as other Windows developers. This chapter shows you some of the ways you can use Visual Studio to keep in touch.

Getting Help

The help documentation is a combination of local offline and online resources, including

- Online and offline Microsoft Developer Network (MSDN) documentation for Visual Studio and .NET
- Online content, such as community forums, from the Microsoft Web site
- Online content from third parties approved by Microsoft to provide content

Book II, Chapter 1 explains configuring online and offline help for Visual Studio.

MSDN includes a library full of documentation, as well as a Knowledge Base. A paid subscription to MSDN allows you to download software for development purposes. For more information about MSDN subscriptions, see Book I, Chapter 5.

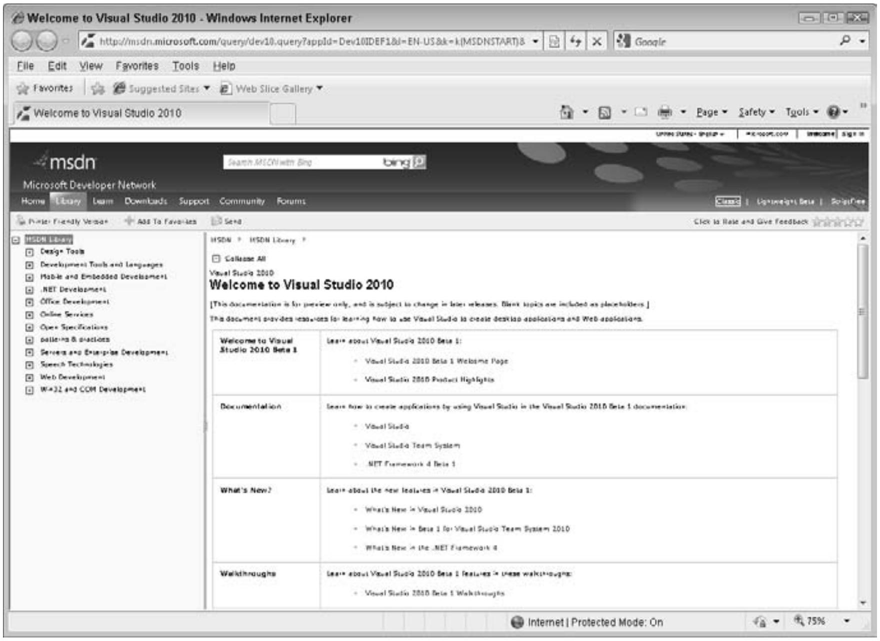
Opening and navigating the help documentation

You can access the Visual Studio help documentation in several ways, including the ones in this list:

- Use the Help menu in Visual Studio.
- Use the features on the MSDN Forums menu.

In Visual Studio 2010 Microsoft has eliminated the Microsoft Document Explorer and instead leverages the web browser that has become a standard paradigm for browsing and searching information.

You also filter the help contents and index by using categories such as .NET Framework, Web development, or a specific .NET language, as shown in Figure 3-1.


Welcome to Visual Studio 2010 - Windows Internet Explorer
http://msdn.microsoft.com/query/dev10.query?appId=Dev10IDEF1&l=EN-US&k=k(MSDNSTART)8
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Welcome to Visual Studio 2010
Page
Safety
Tools
msdn
Microsoft Developer Network
Search MSDN with Bing
bing
Home Library Learn Downloads Support Community Forums
Classic | Lightweight Beta | ScriptFree
Printer Friendly Version
Add To Favorites
Send
Click to Rate and Give Feedback
MSDN Library
Design Tools
Development Tools and Languages
Mobile and Embedded Development
.NET Development
Office Development
Online Services
Open Specifications
patterns & practices
Servers and Enterprise Development
Speech Technologies
Web Development
Win32 and COM Development
MSDN > MSDN Library >
Collapse All
Internet | Protected Mode: On
75%

Visual Studio 2010

Welcome to Visual Studio 2010

[This documentation is for preview only, and is subject to change in later releases. Blank topics are included as placeholders.]

This document provides resources for learning how to use Visual Studio to create desktop applications and Web applications.

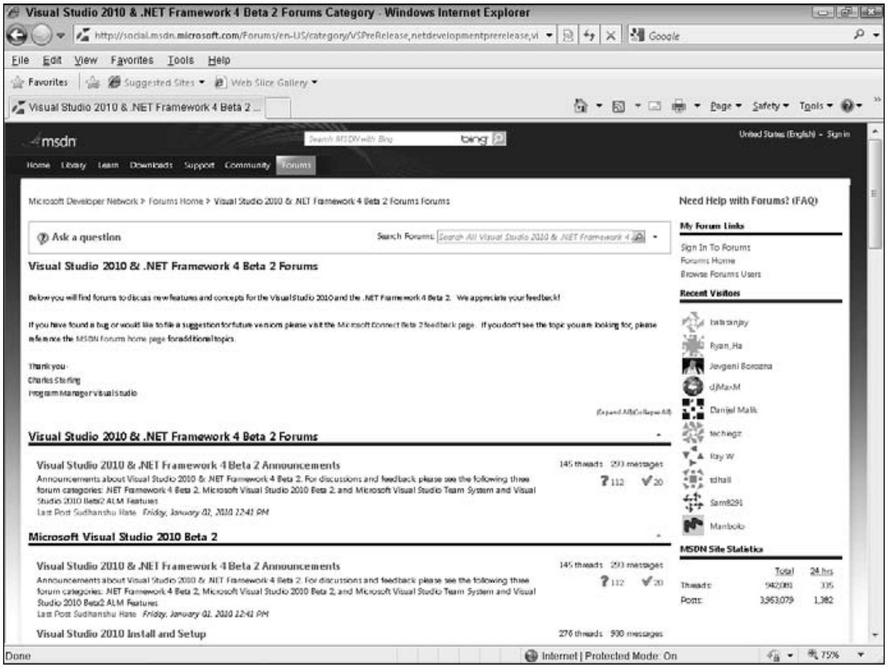
Welcome to Visual Studio 2010 Beta 1	Learn about Visual Studio 2010 Beta 1: - Visual Studio 2010 Beta 1 Welcome Page - Visual Studio 2010 Product Highlights
Documentation	Learn how to create applications by using Visual Studio in the Visual Studio 2010 Beta 1 documentation: - Visual Studio - Visual Studio Team System - .NET Framework 4 Beta 1
What's New?	Learn about the new features in Visual Studio 2010 Beta 1: - What's New in Visual Studio 2010 - What's New in Beta 1 for Visual Studio Team System 2010 - What's New in the .NET Framework 4
Walkthroughs	Learn about Visual Studio 2010 Beta 1 features in these walkthroughs: - Visual Studio 2010 Beta 1 Walkthroughs

Figure 3-1: Filter help contents and the index to narrow the choices.

The Add To Favorites feature creates a bookmark to your favorite help topics. To add a favorite help topic, right-click the topic and choose Add to Help Favorites from the shortcut menu.

The following tools across the top of the window are also helpful when searching for an elusive answer to a question:

- Use special **How Do I?** topics covering everything from Crystal Reports to Visual Studio Tools for Office.
- The **Forums** feature enables you to search Microsoft online forums, ask questions, and check for answers. Figure 3-2 shows the Ask a Question feature, where you can search public forums for answers. You can post questions to the forums if you can't find the information you need.


Visual Studio 2010 & .NET Framework 4 Beta 2 Forums Category - Windows Internet Explorer
http://social.msdn.microsoft.com/Forums/en-US/category/VSPreRelease,netdevelopmentprerelease,vi
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Visual Studio 2010 & .NET Framework 4 Beta 2 ...
Page
Safety
Tools
msdn
Search MSDN with Bing
bing
United States (English) - Sign in
Home Library Learn Downloads Support Community Forums
Done
Internet | Protected Mode: On
75%

Microsoft Developer Network > Forums Home > Visual Studio 2010 & .NET Framework 4 Beta 2 Forums Forums

Ask a question Search Forums: Search All Visual Studio 2010 & .NET Framework 4

Visual Studio 2010 & .NET Framework 4 Beta 2 Forums

Below you will find forums to discuss new features and concepts for the Visual Studio 2010 and the .NET Framework 4 Beta 2. We appreciate your feedback!

If you have found a bug or would like to file a suggestion for future versions please visit the Microsoft Connect Beta 2 feedback page. If you don't see the topic you are looking for, please reference the MSDN Forum home page for additional topics.

Thank you
Charles Sterling
Program Manager Visual Studio

(Expand All/Collapse All)

Visual Studio 2010 & .NET Framework 4 Beta 2 Forums

Visual Studio 2010 & .NET Framework 4 Beta 2 Announcements — 145 threads 293 messages ? 112 ✓ 20
Announcements about Visual Studio 2010 & .NET Framework 4 Beta 2. For discussions and feedback please see the following three forum categories: .NET Framework 4 Beta 2, Microsoft Visual Studio 2010 Beta 2, and Microsoft Visual Studio Team System and Visual Studio 2010 Beta2 ALM Features
Last Post Sudhanshu Hate Friday, January 01, 2010 12:41 PM

Microsoft Visual Studio 2010 Beta 2

Visual Studio 2010 & .NET Framework 4 Beta 2 Announcements — 145 threads 293 messages ? 112 ✓ 20
Announcements about Visual Studio 2010 & .NET Framework 4 Beta 2. For discussions and feedback please see the following three forum categories: .NET Framework 4 Beta 2, Microsoft Visual Studio 2010 Beta 2, and Microsoft Visual Studio Team System and Visual Studio 2010 Beta2 ALM Features
Last Post Sudhanshu Hate Friday, January 01, 2010 12:41 PM

Visual Studio 2010 Install and Setup — 276 threads 930 messages

Need Help with Forums? (FAQ)

My Forum Links
Sign In To Forums
Forums Home
Browse Forums Users

Recent Visitors
babsanjay
Ryan_Ha
Jevgeni Borozna
djMaxM
Danijel Malik
techagz
Ray W
sdhall
Sam8291
Manbolo

MSDN Site Statistics

	Total	24 hrs
Threads:	942,081	335
Posts:	3,953,079	1,382

Figure 3-2: Use the Forums feature to search public forums.

- Use the **Ask a Question button** to ask a question on the MSDN forums.

The Visual Studio help documentation includes topics for .NET, Visual Studio, programming languages, and many other technologies. You can easily get overwhelmed when using the Index or Search features. By synchronizing a topic with the Table of Contents, you can quickly see the domain in which the topic belongs.

See the section about discovering events in Book V, Chapter 4 for an example of using help to locate a topic.

Searching and filtering search results

You can use the Search feature to search the Visual Studio documentation, as well as online content from third-party Web sites. In the Search window, you can filter your results by these criteria:

- **Language:** Visual Basic or C#, for example
- **Technology:** Team System or Windows Forms, for example
- **Content type:** Controls or code snippets, for example

Customizing the start page

Another useful help feature in Visual Studio is the start page. When you open Visual Studio, the start page automatically displays headlines from MSDN. The MSDN headline pages are helpful because they cover topics that developers often need help with. The start page is actually a supplementation to the normal help mechanism.

The news that's displayed on the start page uses really simple syndication (RSS) for displaying news feeds. You can set the news feed on the start page to any RSS feed. For example, to set the start page to display the top stories from CNN, follow these steps:

1. **Choose Tools➪Options in Visual Studio.**
2. **Expand the Environment options by clicking the plus (+) sign in the options tree.**
3. **Click the Startup option.**
4. **Type the URL for top CNN news stories in the Start Page News Channel field:**

   ```
   http://rss.cnn.com/rss/cnn_topstories.rss
   ```

5. **Click OK to save your changes.**

Your start page is updated with the top stories from CNN.

Some developers don't want the RSS information to show when they first run Visual Studio. They may not have an Internet connection, or they may not want to wait for it to update. You can turn the startup page off by choosing Tools➪Options and then making your change on the Startup tab.

Staying in Touch with the .NET Community

Visual Studio puts a major emphasis on using the online community. Part of what makes .NET so valuable is the community that comes along with it. You can use the .NET community to get

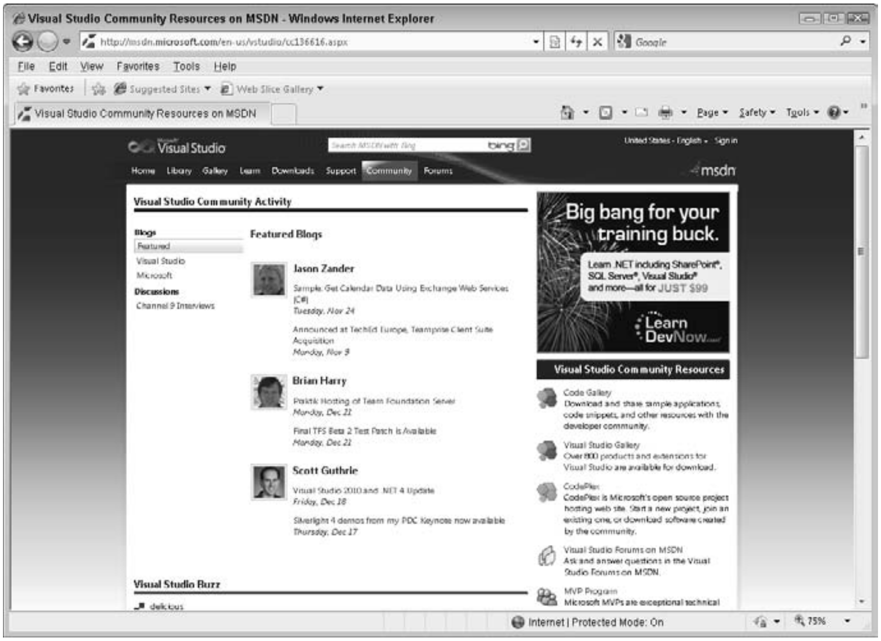
- Help on a topic
- Code samples and starter kits
- Controls and add-ins that extend Visual Studio

The Community menu located on the Visual Studio 2010 documentation page is another means for you to access the .NET community.

Use the Community menu in Visual Studio 2010 help documentation to

- **Ask a question.** Search Microsoft community forums for answers. You can post questions to the forums if you can't find answers to your questions.
- **Check question status.** Check the status of a question you asked in a forum.
- **Send feedback.** Make suggestions and report bugs to Microsoft.
- **Check out the Developer Center.** Visit the Developer Center Web page (on the Microsoft Web site), which provides access to additional Visual Studio resources.
- **Access the Code Gallery.** See links to additional online community Web sites where you can find code samples and tutorials.
- **Access CodePlex**. Visit Microsoft's open source project hosting site to start a new project, join an existing project, or download code shared by the community.
- **Look at the Visual Studio Gallery.** Search an online catalog for tools, components, add-ins, languages, and services that you can buy to extend or use with Visual Studio.
- **Do a community search.** Set the filter on the help search function to one of these five predefined filters:
  - Template and Starter Kits
  - IntelliSense Code Snippets
  - Samples
  - Controls
  - Addins and Macros

Figure 3-3 shows the content in the Community menu of the Visual Studio 2010 documentation.



Visual Studio Community Resources on MSDN - Windows Internet Explorer
http://msdn.microsoft.com/en-us/vstudio/cc136616.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Visual Studio Community Resources on MSDN
Page Safety Tools
Visual Studio
Search MSDN with Bing
United States - English - Sign in
Home Library Gallery Learn Downloads Support Community Forums
msdn
Internet | Protected Mode: On
75%

Visual Studio Community Activity

Blogs
Featured
Visual Studio
Microsoft
Discussions
Channel 9 Interviews

Featured Blogs

Jason Zander
Sample: Get Calendar Data Using Exchange Web Services (C#)
Tuesday, Nov 24
Announced at TechEd Europe, Teamprise Client Suite Acquisition
Monday, Nov 9

Brian Harry
Praktik Hosting of Team Foundation Server
Monday, Dec 21
Final TFS Beta 2 Test Patch is Available
Monday, Dec 21

Scott Guthrie
Visual Studio 2010 and .NET 4 Update
Friday, Dec 18
Silverlight 4 demos from my PDC Keynote now available
Thursday, Dec 17

Visual Studio Buzz
delicious

Big bang for your training buck.
Learn .NET including SharePoint®, SQL Server®, Visual Studio® and more—all for JUST $99
Learn DevNow

Visual Studio Community Resources

Code Gallery
Download and share sample applications, code snippets, and other resources with the developer community.

Visual Studio Gallery
Over 800 products and extensions for Visual Studio are available for download.

CodePlex
CodePlex is Microsoft's open source project hosting web site. Start a new project, join an existing one, or download software created by the community.

Visual Studio Forums on MSDN
Ask and answer questions in the Visual Studio Forums on MSDN.

MVP Program
Microsoft MVPs are exceptional technical

Figure 3-3: Access the Community menu from the Visual Studio 2010 documentation.

Pursuing certification

As you make your way out and about in the .NET community, you'll likely want some way to show that you're not a poser. Microsoft provides a certification series designed to demonstrate, to all who care, that you know your stuff.

Microsoft offers three levels of certification:

- **Technology:** These entry-level certifications demonstrate your proficiency in a specific Microsoft technology, such as .NET Framework, SQL Server, or BizTalk.
- **Professional:** This more-comprehensive set of certifications is geared toward professional software developers and information technology (IT) specialists.
- **Architect:** This peer-reviewed certification requires at least ten years of advanced information technology experience.

Book II
Chapter 3

Making Connections with Visual Studio

Each level of certification accommodates specific certifications. For example, these certifications are available in the Professional series:

- **Microsoft Certified Technology Specialist:** Tests your knowledge of developing software by using .NET and Visual Studio 2010.
- **Microsoft Certified Professional Developer:** Focuses more on specialized job roles, such as Web and Windows developers.
- **Microsoft Certified Applications Developer:** Geared toward people who develop applications by using .NET and Web services.
- **Microsoft Certified Solutions Developer:** Intended for people who build enterprise-level solutions.

To read more about Microsoft certifications, go to

`www.microsoft.com/learning/mcp/newgen`

ON THE WEB

You can get an updated list of resources at the companion Web site (`www.dummies.com/go/vs2010`) for this book.

Viewing Webcasts

In the past few years, the quality of content on MSDN has improved substantially. Not only is the content more accurate, but Microsoft also introduced more kinds of content in addition to plain old documentation. One area that's especially exciting is *Webcasts,* which are free Internet broadcasts of training, tutorials, and presentations.

You can participate in live Webcasts, or you can watch prerecorded Webcasts. An advantage of watching Webcasts live is that you can pose questions to presenters.

You can watch Webcasts about topics ranging from Windows Server 2003 to DotNetNuke to unit testing in Visual Studio 2010. Webcasts are a useful way to dig deeper into the core technologies that you use every day or to get an overview of topics.

Here are some Webcast resources you should check out:

- The MSDN Events and Webcasts page features Webcasts for Visual Studio and SQL Server: `http://msdn.microsoft.com/events`
- The TechNet Events and Webcasts page helps you keep up with technical topics: `www.microsoft.com/technet/community/webcasts`
- View Webcasts related to Visual Studio Team System: `http://msdn2.microsoft.com/teamsystem/aa718763.aspx`

Chapter 4: Upgrading .NET

In This Chapter

- **Deciding when to upgrade**
- **Converting Windows applications and class libraries**
- **Exploring conversion options for Web applications**
- **Using Visual Studio 2010 with previous versions of .NET**

What does it mean to upgrade to Visual Studio 2010? Two tasks are required for an upgrade:

- Move to the Visual Studio 2010 toolset.
- Use version 4.0 of the .NET Framework.

Depending on your situation, you may decide to

- Start using Visual Studio 2010 and .NET 4.0 right away for all new development.
- Upgrade all your existing Windows and Web applications to Visual Studio 2010 and .NET 4.0.
- Leave all or some of your existing applications in the versions of Visual Studio with which they were created.
- Upgrade your existing applications to Visual Studio 2010 while still using a previous version of the .NET Framework.

In this chapter, I talk about these scenarios and walk you through some conversion processes.

Making the Business Case for Upgrading

When you make the decision to upgrade to Visual Studio 2010, your existing applications must be converted to work with Visual Studio 2010 before they can be opened. In most cases, the conversion process doesn't change your code. Instead, it merely updates configuration files, such as project and solution files. The changes that are made depend on whether you're converting

- Windows applications
- Web applications
- Class libraries

Visual Studio 2010 uses version 4.0 of the .NET Framework by default. The syntax of programming languages, such as C# and Visual Basic, may change with each new version of the .NET Framework as new features are added and existing features are improved. The conversion process doesn't update your code to the latest syntax. Rather, the newer versions of .NET continue to support the syntax of previous versions — for example, upgrading versions of .NET doesn't break existing code. It's up to you to figure out whether a new and improved syntax is available and decide whether you want to change your code to use the new syntax. The languages add features in most cases, rather than actually change existing features. In most cases, the changes are slight, rather than major code busting features.

When you make the decision to upgrade to Visual Studio 2010, you have lots of choices. In the sections that follow, you find out more specifics about what happens to code when you upgrade, alternatives to upgrading, and what you need to do to upgrade programs in pre-.NET languages.

What happens to Visual Studio code in an upgrade?

As you think about whether you should upgrade, consider these issues:

- **Windows applications and class libraries aren't changed much by the conversion process.** Even though configuration settings are updated, your code stays the same.
- **Any code or controls that were deprecated in version 4.0 of the .NET Framework still work.** You can't use the visual designers for deprecated controls, although you can use the code editor.
- **Your class libraries aren't optimized to take advantage of the latest features in .NET 4.0.** You have to figure out whether you should change your code.
- **After converting applications to Visual Studio 2010, analyze your code by using FxCop.** It offers you suggestions for improving your code. See Book V, Chapter 5 for more information about FxCop.
- **Web applications require some finesse to get all their components converted properly.** You need to do some preparation work before you convert them.
- **Applications that are especially complex can be difficult to convert.** If you can't convert the entire solution, you may have to convert individual projects one at a time and then add them to your Visual Studio 2010 solution.

- **Clients running your applications must have the .NET 4.0 framework installed.** You can convert your applications to work with Visual Studio 2010 and deploy applications to work with previous versions of the .NET Framework. I show you how in the section “Using Visual Studio 2010 with .NET 2.0,” later in this chapter.
- **You can’t unconvert.** You can restore your backups, however, and discard the converted application if you don’t like the results.

What are alternatives to upgrading?

You have several alternatives to performing an outright upgrade of your application to Visual Studio 2010:

- **If it ain’t broke . . . .** You may not have any compelling reason to upgrade your Visual Studio 2002, 2003, 2005, and 2008 applications to Visual Studio 2010. Unless you plan to take advantage of the new Visual Studio 2010 features in your Visual Studio 2002, 2003, 2005, and 2008 applications, you probably don’t need to upgrade.
- **Go side by side.** You can run all versions of the .NET Framework and Visual Studio on a single computer. Your existing version applications then work alongside any new version 4.0 applications you create.
- **Step up to the 4.0 framework.** Chances are that your .NET applications will run on the 4.0 framework, even without converting them to work with Visual Studio 2010. You can take advantage of the new security and performance features of .NET 4.0 without touching your code.
- **Mix and match.** If you have relatively stable parts of your application that use previous versions of the .NET Framework and you want to do new development in Visual Studio 2010, you can mix and match. Just reference your older components in your new Visual Studio 2010 solution.
- **Take the brute-force approach.** You can manually re-create your applications in Visual Studio 2010 by choosing to copy and paste your old code into newly created Visual Studio 2010 solutions.

Indulge in the best of both worlds. You can target the .NET 2.0 and 3.5 runtimes with your existing applications in Visual Studio 2010. I show you how in the “Using Visual Studio 2010 with .NET 2.0,” later in this chapter.

How do you upgrade from pre-.NET languages?

This chapter assumes that you’re converting from a previous version of the .NET Framework to the current version. If you’re upgrading from a pre-.NET language, such as Visual Basic 6, then you’ll need to jump through a few hoops to get your application converted. After they’re converted, you’ll still likely need to finesse the converted code to make it work.

You may be able to reuse stable components created in Visual Basic 6.0 by using a feature of the .NET Framework called COM Interop. With COM Interop, you can consume your Visual Basic 6 code from your .NET applications.

In the case of ASP.NET, you have additional considerations when you choose to upgrade to Visual Studio 2010 and ASP.NET 4.0:

- **ASP.NET uses a new, simplified model for managing resources, such as code and Web pages.** Project files are no longer needed because all the configuration settings are moved to `web.config`, an XML-based configuration file.

  If you're really attached to using project files, you can use the Web Application Projects extension for Visual Studio 2005. Download the extension for free from Microsoft's Web site at `http://msdn2.microsoft.com/asp.net/aa336618.aspx`. This extension still works with Visual Studio 2010.

- **You can execute your Web sites on local development machines by using the ASP.NET Developer Server.** Using the ASP.NET Developer Server requires less configuration and is more secure than using Internet Information Services (IIS) as your local Web server.
- **ASP.NET 4.0 offers a new code-behind model, new controls for laying out Web pages and working with data, and improved deployment options.** Your productivity is greatly improved with ASP.NET 4.0.

See Book III for more information on creating Web applications with ASP.NET 4.0.

Considering Your Conversion Strategies

Visual Studio provides a conversion wizard for migrating projects from previous versions of .NET to the .NET 4.0 version. The conversion process itself is fairly straightforward: You just open your Visual Studio 2002, 2003, 2005, or 2008 solution in Visual Studio 2010, and the Conversion Wizard automatically opens and walks you through the conversion process.

Before you convert any applications, you need to do some preparation work on all of them:

- Make sure that you have good backups, in case you decide to roll back to the unconverted version.
- Make sure that all your source code compiles before you convert.

- Take the time to clean up any extraneous files that don't belong with the application.
- Make sure that the .NET 4.0 Framework is installed on all client machines that will run your application.

Converting Web applications requires more preparation than converting Windows applications because the conversion process is more involved. See "Converting Web applications," later in this chapter, for more details.

Converting Windows applications and class libraries

As a rule, you should try to convert an entire solution first rather than convert each of the projects in a solution individually. If the conversion process can't convert all the projects in your solution, you should

- Allow the Visual Studio Conversion Wizard to convert the solution and as many projects as it can.
- Open individual projects that don't convert in Visual Studio 2010. Walk through the Conversion Wizard for each project that doesn't convert with the solution.
- Add the converted projects to the converted solution.

To convert Windows solutions and projects, follow these steps:

1. **In Visual Studio 2010, choose File➪Open➪Project/Solution.**

   The Open Project window appears.

2. **Browse to your Visual Studio 2002, 2003, 2005, or 2008 Windows application and click the Open button.**

   If you have a solution file, you should open it.

   The Visual Studio Conversion Wizard appears.

3. **Click Next to start the wizard.**

4. **Click Select the radio button next to Yes, Create a Backup before Converting radio button, specify a backup location, and then click Next.**

TIP

It's a good idea to create a backup of your existing solution file in case something goes wrong in the conversion. If something goes awry, you can revert back to your backup copy. After you upgrade your solution file, you can no longer use it in previous versions of Visual Studio.

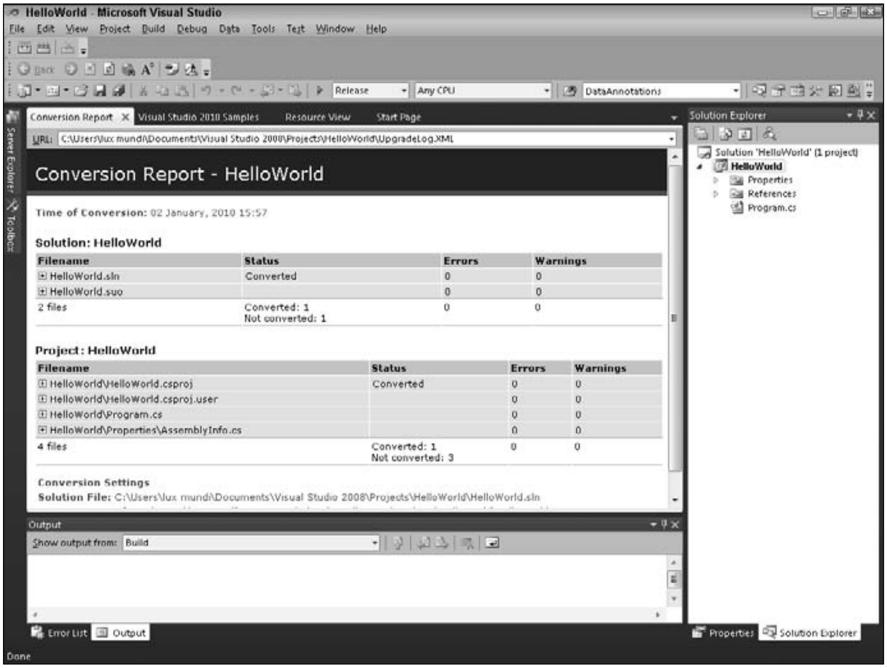
A summary of the solution and projects to be converted is displayed.

5. **Click the Finish button to start the conversion.**

   After the conversion is done, a completion window appears. Any errors that occur during the conversion process are listed.

6. **To display a conversion log, select the Show the Conversion Log When the Wizard Is Closed option and then click the Close button.**

   The conversion log, shown in Figure 4-1, appears in Visual Studio 2010.



URL: C:\Users\lux mundi\Documents\Visual Studio 2008\Projects\HelloWorld\UpgradeLog.XML

Conversion Report - HelloWorld

Time of Conversion: 02 January, 2010 15:57

Solution: HelloWorld

Filename	Status	Errors	Warnings
⊞ HelloWorld.sln	Converted	0	0
⊞ HelloWorld.suo		0	0
2 files	Converted: 1 Not converted: 1	0	0

Project: HelloWorld

Filename	Status	Errors	Warnings
⊞ HelloWorld\HelloWorld.csproj	Converted	0	0
⊞ HelloWorld\HelloWorld.csproj.user		0	0
⊞ HelloWorld\Program.cs		0	0
⊞ HelloWorld\Properties\AssemblyInfo.cs		0	0
4 files	Converted: 1 Not converted: 3	0	0

Conversion Settings

Solution File: C:\Users\lux mundi\Documents\Visual Studio 2008\Projects\HelloWorld\HelloWorld.sln

Figure 4-1: View the conversion log for details about the conversion process.

Follow these steps again to convert class libraries.

After the conversion, build your application in Visual Studio 2010 to make sure that all converted projects are compiled. If you have multiple projects in your solution, you should build your class libraries first and work your way through any dependencies until all your components are built.

Converting Web applications

The conversion process for Web applications is more involved than for Windows applications and class libraries. The Visual Studio Conversion Wizard performs several additional functions when converting Web applications, including the following:

- Move project file configuration settings into the `web.config` file.
- Create new folders, such as `App_Code`, and move stand-alone class files.
- Update the code behind attributes to reflect new attributes.

Because the Web application conversion process is more involved, the preparation requires that you do a few more preparation tasks as well:

- **Make sure that all files that need to be converted are included in the project.** The Visual Studio Conversion Wizard converts only those files.

  To include in a project any files that were previously excluded, open the project in Visual Studio 2002, 2003, 2005, or 2008, right-click the file, and then choose Include in Project from the contextual menu.
- **Remove any files from your project directory that aren't part of the project.**
- **When you convert multiple projects, make sure that no overlap occurs in the files that each project references.**
- **When you convert multiple solutions, make sure that no overlap occurs in the projects contained in the solutions.**
- **Remove references to other Web projects.**

To convert a Web application, follow these steps:

1. **Choose File➪Open➪Web Site.**

   The Open Web Site window appears.

   You must open your Web site using the Open Web Site window in order for the Conversion Wizard to work properly.

   The Open Web Site window gives you several sources from which you can open your Web site. In most cases, you'll open your site from a local IIS site. You can also choose to open a site from a File Transfer Protocol (FTP) site, remote Web server, or source control.

   Don't select File System as your source. Converting a Web application from a file system causes the Web site to lose data about the site's structure.
2. **Select Local IIS from the list of sources in the left pane.**

   A list of Web sites on the local computer appears.
3. **Navigate to the Web site you want to convert, select it (as shown in Figure 4-2), and then click the Open button.**

   The Visual Studio Conversion Wizard appears.

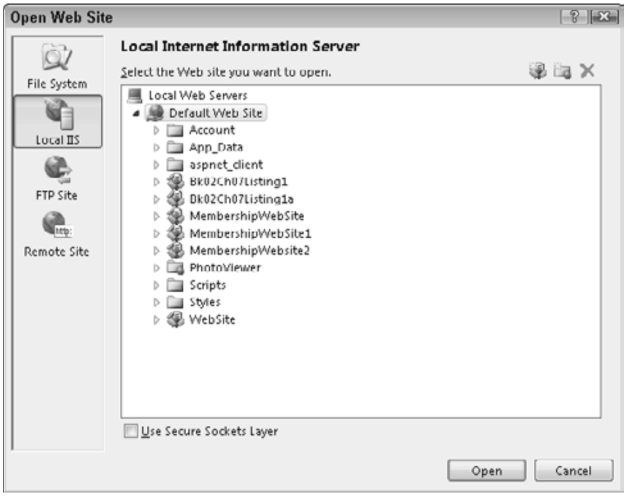


Figure 4-2: Open an existing Web site by using the Local IIS option.

4. **Click Next to start the wizard.**
5. **Follow the steps in the Visual Studio Conversion Wizard to complete the conversion.**

If errors occur in the conversion, read the conversion log and determine which files weren't converted. Remember that class files aren't converted. They're copied to the solution directory as-is.

You may check the `ConversionReport.webinfo` file to view the steps taken to convert the application. The file is named `ConversionReport.txt` if you haven't installed the Conversion Wizard upgrade.

Build your applications by making sure that all dependent projects are built first. Always open your converted project by using the Local IIS option.

After the conversion, you have an ASP.NET 4.0 Web application that you can edit in Visual Studio 2010. The folder structure and code-behind model are updated to the new ASP.NET 4.0 model.

Running Web applications side-by-side

You can specify which version of the .NET Framework you want a Web application to use. Version 4.0 of the framework is backward-compatible with versions 2.0, 3.0, and 3.5. You can then run your existing applications in version 4.0 and take advantage of improved security and performance features without recompiling.

Configuring your Web application to use version 4.0 of the .NET Framework requires that the .NET Framework version 4.0 is installed on the server where the application is hosted. If that version isn't installed, browse to the Windows Update site on the server and install it.

To configure a Web application to use version 4.0 of the .NET Framework, follow these steps:

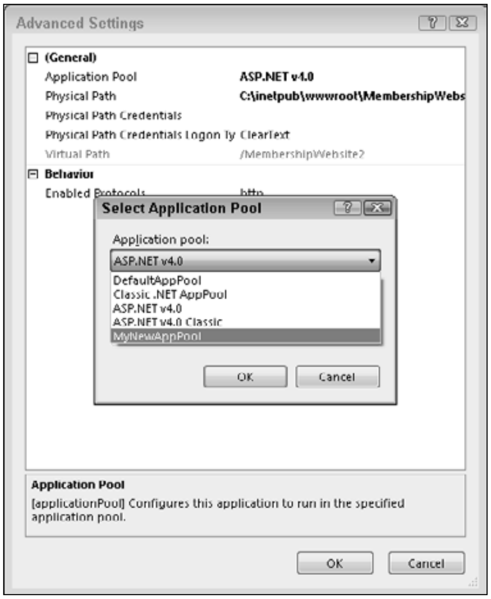
1. **Launch the Internet Information Services Manager on the server.**

   In Windows 7, you can find the IIS Manager by choosing Administrative Tools in the Control Panel.

2. **Select View Application Pools item.**
3. **Right-click any items under the Name column.**

   A menu appears.

4. **Select Basic Settings.**
5. **Set the version by using the ASP.NET version drop-down list, as shown in Figure 4-3.**
6. **Click OK.**



Advanced Settings

(General)
Application Pool: ASP.NET v4.0
Physical Path: C:\inetpub\wwwroot\MembershipWebs
Physical Path Credentials
Physical Path Credentials Logon Ty: ClearText
Virtual Path: /MembershipWebsite2

Behavior
Enabled Protocols: http

Select Application Pool

Application pool: ASP.NET v4.0

Application Pool
[applicationPool] Configures this application to run in the specified application pool.

Figure 4-3: Set the version of the .NET Framework you want the site to use.

If you like to take control of your conversions, you can re-create your applications manually in Visual Studio 2010. Here's the overview:

1. **Create new Windows or Web solutions.**
2. **Use the visual designers to manually re-create your forms, pages, user controls, and any other elements that are in your existing application.**

   Make sure to keep your naming style consistent with your old application.
3. **Copy any code that you want to bring over from your old solution.**

   Your code should work as long as you re-created all your Windows and Web forms properly.
4. **Build your applications and deal with errors.**

Using Visual Studio 2010 with .NET 2.0

Suppose that you want the best of all worlds. You want to take advantage of the exciting new features in the Visual Studio 2010 development environment, such as the class designer and code snippets, and you still need to target the .NET 2.0 runtime environment. Well, my friend, that option is available to you.

Note that you can't use any of the features of the .NET 4.0 framework, such as lambda functions, LINQ, and master pages. If you try, you receive an error message when you build your solution. Also, you can't use this trick with Web applications. ASP.NET uses a completely different build model than Windows applications and class libraries do. You can still get the lay of the Visual Studio 2010 land, though, while continuing to target previous versions of the .NET Framework.

See Book III for more information about master pages and Book V for more information about working with generics.

To target a previous version of .NET 2.0 with your application, follow these steps:

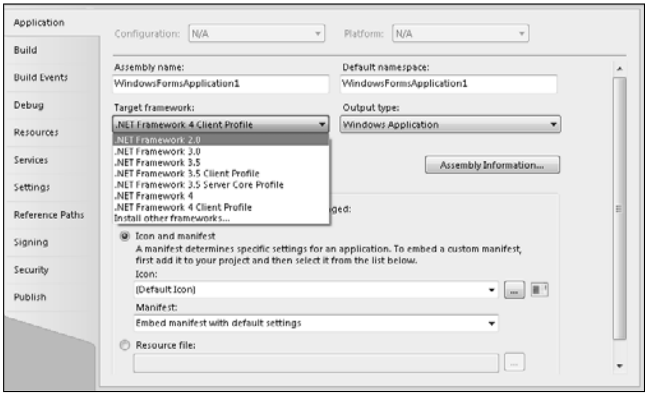
1. **Open your Visual Basic or C# application in Visual Studio 2010 by using the File menu.**

   If you're opening an application created in a previous version of Visual Studio, you need to walk through the Conversion Wizard steps before you can open the application.

2. **Choose Tools➪Options and then click the Application tab.**

   The Options dialog box appears.

3. **Choose .NET Framework 2.0 from the Target Framework drop-down list, shown in Figure 4-4.**



Configuration: N/A
Platform: N/A
Assembly name: WindowsFormsApplication1
Default namespace: WindowsFormsApplication1
Target framework: .NET Framework 4 Client Profile
Output type: Windows Application

● Icon and manifest
A manifest determines specific settings for an application. To embed a custom manifest, first add it to your project and then select it from the list below.
Icon: (Default Icon)
Manifest: Embed manifest with default settings
○ Resource file:

Figure 4-4: Select .NET 2.0 from the list of available platforms.

4. **Click OK.**

   The platform for your project is updated to .NET 2.0 in the Configuration Manager. When you build your application, Visual Studio 2010 targets the .NET 2.0 version of the .NET Framework.

5. **Click the Close button to close the Configuration Manager.**

6. **Build your application by choosing Build➪Build Solution.**

The Microsoft team responsible for MSBuild is also working on its own tool for using Visual Studio 2010 with the .NET 2.0 compiler. Its project is named MSBee, which stands for MSBuild Everett Environment. MSBee works similarly to the MSBuild Toolkit. The main difference is that you can't use MSBee using the Visual Studio 2010 Build menu. Instead, you have to manually build your applications using the command line. You can download MSBee for free at `www.codeplex.com/Wiki/View.aspx?ProjectName=MSBee`.

You aren't restricted to using these two applications. The `.targets` files are what makes it possible to create .NET 2.0, .NET 3.0, and .NET 3.5 assemblies using Visual Studio 2010. You can browse the Web for `.targets` files or even write your own.

Book III

Coding

The 5th Wave By Rich Tennant

"We're here to clean the code."

Contents at a Glance

Chapter 1: Programming with Visual Studio 2010

In This Chapter

- Performing common tasks in the Code Editor
- Organizing code in C# and Visual Basic
- Getting started with the basics of C# and Visual Basic

Do you ever get the feeling that sometimes you're not alone? If you've ever written a program with one of the Visual Studio Code Editors, you know what I mean. Writing code in Visual Studio is like having a word processor that knows what you want to say next.

When you start to type code in Visual Studio, helpers appear magically out of nowhere. Like the host at your favorite restaurant, they share with you a list of today's specials: "Might I suggest an `Integer`, or how about a `DataSet`, sir?"

Even if you have no clue about all the choices that Visual Studio offers in the convenient pop-up list, just hover your mouse over one of them, and a description appears. Like a museum tour guide, Visual Studio shares with you the intimate details of the programming artifacts you're inspecting.

This chapter shows you some of the magical helpers and guides that Visual Studio uses to help you write better code. You also get a quick overview of how to write programs using two popular .NET languages: C# and Visual Basic.

Using the Code Editor

Although Visual Studio provides a great number of visual designers that allow you to generate code by dragging and dropping controls and components onto visual designers, at some point, you have to write code. Visual Studio really helps you write code by providing you with many editors for editing all kinds of files. The Source Code Editor is especially feature rich, offering a number of productivity boosting features.

Here are some of the ways to open source code files in the Code Editor:

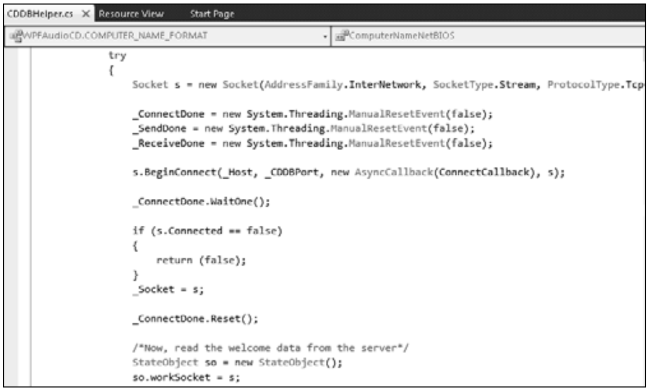
- Double-click a source code file in Solution Explorer.
- Press F7 while viewing a form in the Windows or Web Forms designers.
- Add a new source code file via the Add New Items window.

The Code Editor is more than just a plain text editor like Notepad. You can use the Code Editor to do many things, such as

- Enter code in the main Code Pane area.
- Set breakpoints in the gray Indicator Margin along the left of the editor.
- Select lines of code by clicking the space between the Code Pane and the Indicator Margin.

  You can optionally display line numbers in the Selection Margin.
- Collapse code by using the outline lines along the left margin of the Code Editor.
- Jump to code via the Navigation bar along the top of the Code Editor.

Figure 1-1 shows an example of the Code Editor.



```
        try
        {
            Socket s = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp

            _ConnectDone = new System.Threading.ManualResetEvent(false);
            _SendDone = new System.Threading.ManualResetEvent(false);
            _ReceiveDone = new System.Threading.ManualResetEvent(false);

            s.BeginConnect(_Host, _CDDBPort, new AsyncCallback(ConnectCallback), s);

            _ConnectDone.WaitOne();

            if (s.Connected == false)
            {
                return (false);
            }
            _Socket = s;

            _ConnectDone.Reset();

            /*Now, read the welcome data from the server*/
            StateObject so = new StateObject();
            so.workSocket = s;
```

Figure 1-1: Use the Code Editor to write source code.

Simplifying your common tasks

The Code Editor has many features similar to what you'd expect to find in a word processor. Some features, such as formatting, however, are slightly different. For example, you can't bold your code in the Code Editor, but you can use familiar features, such as copy and paste.

Managing indents

An important aspect of writing code is handling indentation. When you properly indent, you make it easier to read on-screen. The Visual Studio Code Editor provides three indentation styles:

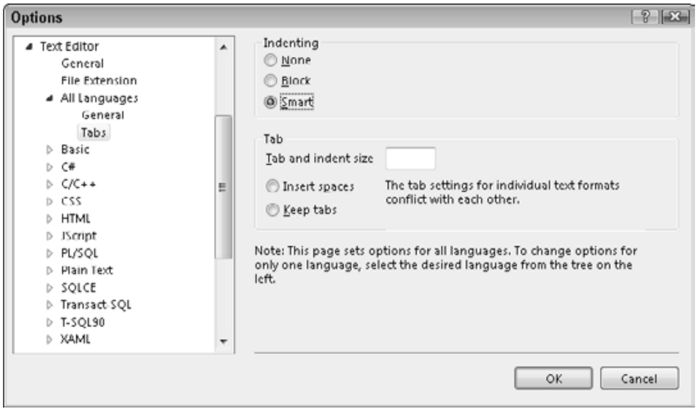
- **None:** No indentation is applied to code.
- **Block:** Indentation matches the preceding line of code.
- **Smart:** Lines are automatically indented according to the standards of the programming language.

By default, the Code Editor indents each code line based on the standards for the programming language: *smart indenting.* When you're typing your code, the Editor automatically indents the line. When smart indentation is enabled, you can't manually increase indentation by pressing the Tab key.

To change the indentation style used by the Code Editor, follow these steps:

1. **Choose Tools➪Options.**
2. **Expand the Text Editor folder.**
3. **Expand a language to configure or click the All Languages folder to apply the setting to all programming languages.**
4. **Click Tabs.**

   A list of tab settings appears.
5. **Set the Indenting style, as shown in Figure 1-2, and then click OK.**



Indenting: ○ None ○ Block ● Smart

The tab settings for individual text formats conflict with each other.

Note: This page sets options for all languages. To change options for only one language, select the desired language from the tree on the left.

Figure 1-2: Set the indentation style for the Code Editor.

Accessing formatting commands

You can access formatting features to indent your code by choosing Edit➪Advanced. Table 1-1 lists the Code Editor's formatting commands and what they do. Note that not all commands are available for all languages.

Table 1-1 Code Editor Formatting Commands

Menu Item	*Description*
Format Document	Applies indentation style to the entire document
Format Selection	Applies indentation style to the selected text
Tabify Selected Lines	Converts indentations to tab characters
Untabify Selected Lines	Converts indentations to white space
Make Uppercase	Changes the code of the selected text to uppercase
Make Lowercase	Changes the code of the selected text to lowercase
Delete Horizontal White Space	Deletes all white space on the line
View White Space	Displays character marks in the editor
Word Wrap	Toggles line wrapping on and off
Incremental Search	Activates a search of your document, letter by letter, as you type
Comment Selection	Places comment characters in front of the selected text
Uncomment Selection	Removes comment characters from selected text
Increase Line Indent	Increases indentation
Decrease Line Indent	Decreases indentation

Some indentation and comment commands are available on the Text Editor toolbar, as shown in Figure 1-3.

Most of these commands are useless if you're using smart indentation. Smart indentation automatically manages indentation for you. You might choose to turn off smart indentation if your company uses a different indentation standard. Also, you might want to use your own indentation standard, such as using white spaces instead of tabs.

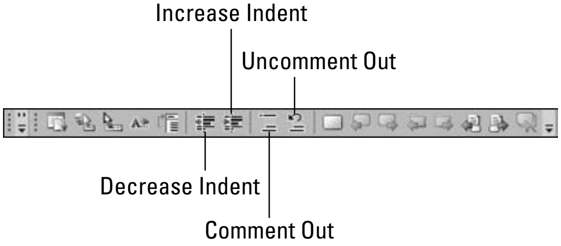


Figure 1-3: Use the Text Editor toolbar to increase indentation or comment out a selection.

Searching with Find and Replace

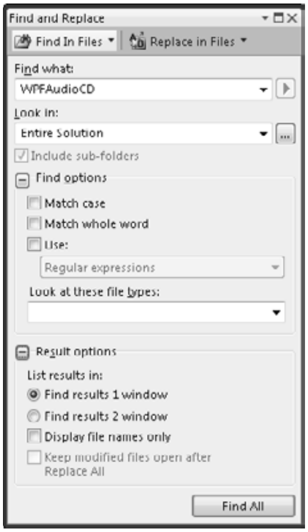
Visual Studio provides extensive Find and Replace capabilities that go way beyond what you find in a standard text editor. Table 1-2 lists the Find and Replace options. You access the Find and Replace commands by choosing Edit➪Find and Replace or by using keyboard shortcuts.

Table 1-2 Find and Replace Commands

Command	*Keyboard Shortcut*	*Purpose*
Quick Find	Ctrl+F	Searches for text
Quick Replace	Ctrl+H	Replaces text
Find in Files	Ctrl+Shift+F	Includes options for searching for text in a set of files
Replace in Files	Ctrl+Shift+H	Includes options for replacing text in a set of files
Find Symbol	Alt+F12	Restricts search scope to symbols only

The Find and Replace commands all use the same dialog box, and most of the commands use the same options. As Figure 1-4 shows, your Find and Replace options are the following:

- **Find What:** In this text box, enter text for which to search.
- **Replace With:** In this text box, enter text to replace the found text.
- **Look In:** From this drop-down list, specify the scope of the search, such as Current Document or Entire Solution.



Find what: WPFAudioCD

Look in: Entire Solution

☑ Include sub-folders

Find options

□ Match case

□ Match whole word

□ Use: Regular expressions

Look at these file types:

Result options

List results in:

● Find results 1 window

○ Find results 2 window

□ Display file names only

□ Keep modified files open after Replace All

Figure 1-4: Use Find and Replace commands to search your code.

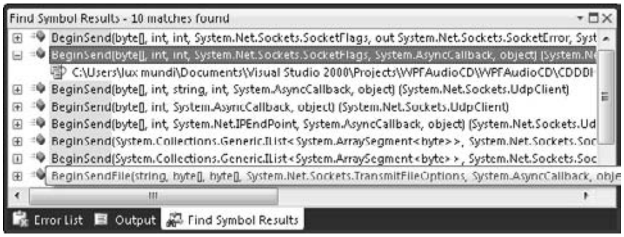
- ✦ **Find Options:** Use these checkboxes to specify options, such as whether to match case or the whole word.
- ✦ **Find Next:** Click this button to find the next text match. The button toggles to **Replace Next** when you are in Replace mode.
- ✦ **Replace All:** Clicking this button replaces all matches. The button toggles to **Find All** when you are in Find mode.

To select a different kind of Find and Replace command, open the drop-down lists displayed at the top of the Find and Replace dialog box.

TIP

Before you use Find and Replace, ask yourself whether one of the Visual Studio new refactoring features might work better. See Chapter 5 of Book III for an overview of refactoring in Visual Studio.

The Find Symbols command restricts your search to only symbols. *Symbols* are the methods and type definitions declared in source code. You can use the Find Symbols command to search your own code or external libraries for which you don’t have source code. The results of the Find Symbols code appear in the Find Symbol Results window. Figure 1-5 shows the results from a search for `BeginSend` in a set of source code. The results show exactly where `BeginSend` is used in source code. Double-click any of the results to go to the source code.



Find Symbol Results - 10 matches found

```
DeginSend(byte[], int, int, System.Net.Sockets.SocketFlags, out System.Net.Sockets.SocketError, Syst
BeginSend(byte[], int, int, System.Net.Sockets.SocketFlags, System.AsyncCallback, object) (System.Ne
    C:\Users\lux mundi\Documents\Visual Studio 2000\Projects\WPFAudioCD\WPFAudioCD\CDDDI
BeginSend(byte[], int, string, int, System.AsyncCallback, object) (System.Net.Sockets.UdpClient)
BeginSend(byte[], int, System.AsyncCallback, object) (System.Net.Sockets.UdpClient)
BeginSend(byte[], int, System.Net.IPEndPoint, System.AsyncCallback, object) (System.Net.Sockets.Ud
BeginSend(System.Collections.Generic.IList<System.ArraySegment<byte>>, System.Net.Sockets.Soc
BeginSend(System.Collections.Generic.IList<System.ArraySegment<byte>>, System.Net.Sockets.Soc
BeginSendFile(string, byte[], byte[], System.Net.Sockets.TransmitFileOptions, System.AsyncCallback, obje
```

Figure 1-5: The Find Symbol Results window displays all the instances where the text appears in symbols.

Getting on task

Use the Visual Studio Task List to keep track of a simple list of to-do items. The Task List window displays two kinds of tasks:

- **User tasks:** Tasks entered directly into the Task List window
- **Task comments:** Comments placed in your source code using special tags

The Task List provides a consolidated list of tasks you enter directly into the Task List window and tagged comments you place inside your source code. The default set of tags you use to mark your comments comprises `TODO`, `HACK`, or `UNDONE`.

You can create your own custom tags.

To tag a comment within your source code, use the standard comment mark for your programming language, followed by one of the task comment tags. For example, to add a task comment in Visual Basic or C#, you type the following:

VB:

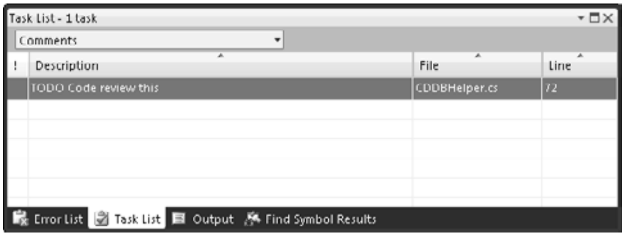
```
'todo Code review this
```

C#:

```
// todo Code review this
```

Tags may be entered in upper- or lowercase. By default, tags are in uppercase.

The Task List window consolidates all your task comments in a single view. You can double-click a comment in the Task List to jump to the location in the source code where the comment appears. To open the Task List window, click Task List on the View window. Figure 1-6 shows an example of the Task List window.



!	Description	File	Line
	TODO Code review this	CDDBHelper.cs	72

Figure 1-6: Task comments from source code appear in the Task List window.

TIP

Always check the Task List for code samples you download from Microsoft and other third parties. The samples usually include a list of tasks you need to complete in order to use the sample.

You can add your own task comment tags. Figure 1-6 uses a comment tag called `TODO`. To add your own task comment tags, do the following:

1. **Choose Tools➪Options.**
2. **Expand the Environment folder.**
3. **Click Task List.**
4. **Type a name for the comment tag in the Name text box.**
5. **Click the Add button.**

   Your new comment tag appears in the list.
6. **Click OK.**

Collapsing code

The Code Editor includes outlining features you can use to expand and collapse your source code. Use the Edit➪Outlining menu to access options for toggling outlining options.

The .NET programming languages provide syntax to create a named section of code you can collapse and expand, known as a *region directive.* To use a region directive, sandwich your code block between a start and end region directive. The syntax for the directive varies, depending on the language you're using.

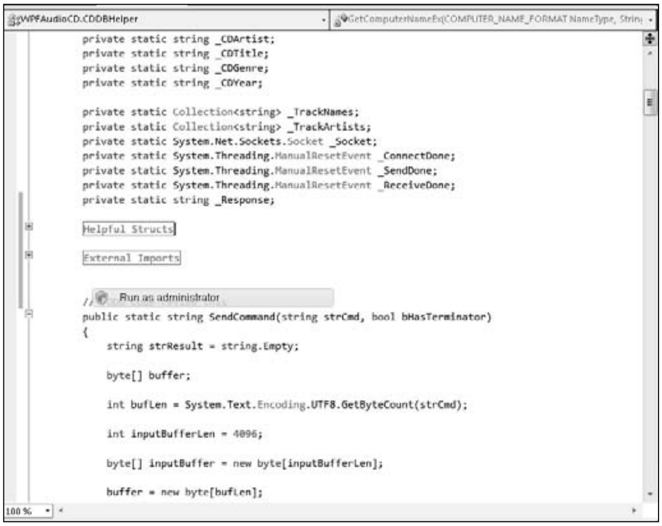
In Visual Basic, the directive looks like this:

```
#Region "Description of code block"
'source code here
#End Region
```

In C#, the directive appears as follows:

```
#region Description of code block
//source code here
#endregion
```

Note that C# doesn't require double quotes around the description. Figure 1-7 shows an example of a class using regions to organize the source code.



```
        private static string _CDArtist;
        private static string _CDTitle;
        private static string _CDGenre;
        private static string _CDYear;

        private static Collection<string> _TrackNames;
        private static Collection<string> _TrackArtists;
        private static System.Net.Sockets.Socket _Socket;
        private static System.Threading.ManualResetEvent _ConnectDone;
        private static System.Threading.ManualResetEvent _SendDone;
        private static System.Threading.ManualResetEvent _ReceiveDone;
        private static string _Response;

        Helpful Structs

        External Imports

        //
        public static string SendCommand(string strCmd, bool bHasTerminator)
        {
            string strResult = string.Empty;

            byte[] buffer;

            int bufLen = System.Text.Encoding.UTF8.GetByteCount(strCmd);

            int inputBufferLen = 4096;

            byte[] inputBuffer = new byte[inputBufferLen];

            buffer = new byte[bufLen];
```

Figure 1-7: Use a region directive to create collapsible blocks of code.

Using IntelliSense

The Visual Studio Code Editor has a special feature — IntelliSense — that helps you complete syntax while you type. You might recognize IntelliSense as the feature that brings your dot notation to life. When you type a variable's name and then type a period or dot, you get a list of properties and methods that you can use for that variable. The context-aware list of properties and methods is brought to you courtesy of IntelliSense.

IntelliSense is capable of more than listing methods and properties. You can see IntelliSense in action when you

- ✦ Hover your mouse over a property or method to view its signature.
- ✦ Open the parentheses on a method and receive feedback about acceptable parameter values.

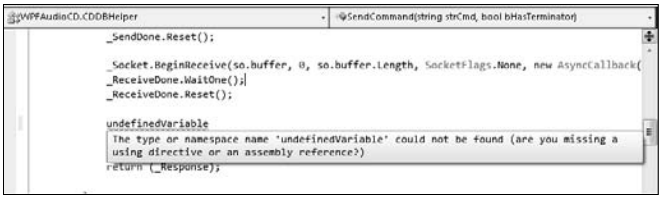
The new IntelliSense features in Visual Studio 2010 take point-and-click code generation to a whole new level. See Chapter 5 in Book III for a demonstration.

Using visual cues

The Visual Studio Code Editor provides several visual cues that you can use while writing code, such as these:

- ✦ **Colors are used to identify different kinds of code.** For example, strings are red, and commented text is green.
- ✦ **Bookmark and breakpoint symbols appear in the far-left margin.**
- ✦ **Coding errors have colored squigglies beneath them to indicate the kind of error.** For example, syntax errors appear with red squigglies beneath them, whereas warnings use green squigglies beneath them.

Figure 1-8 shows an example of a squiggly underline on an undeclared variable.



```
            _SendDone.Reset();

            _Socket.BeginReceive(so.buffer, 0, so.buffer.Length, SocketFlags.None, new AsyncCallback(
            _ReceiveDone.WaitOne();
            _ReceiveDone.Reset();

            undefinedVariable
            The type or namespace name 'undefinedVariable' could not be found (are you missing a
            using directive or an assembly reference?)
            return (_Response);
```

Figure 1-8: The Code Editor uses visual cues, such as squiggly lines, for errors. You can also turn line numbers on.

You can optionally turn on line numbers to help you with your code. To turn on line numbers, follow these steps:

1. **Choose Tools➪Options.**
2. **Expand Text Editor.**

3. **Click All Languages.**
4. **Select the Line Numbers check box.**

Use the options in the Text Editor section to customize the Code Editor. You may want to disable certain features in the Code Editor, such as word wrap, if you're working with especially large source code files. Disabling features when you're editing large files can make the editor less sluggish.

Browsing and navigating code

Often, you need to browse your own code or try to navigate your way through code that someone else has written. Visual Studio provides two basic kinds of tools for browsing code:

- When you have the source code
- When you have a compiled assembly

The Code Editor has the following tools for browsing your own source code:

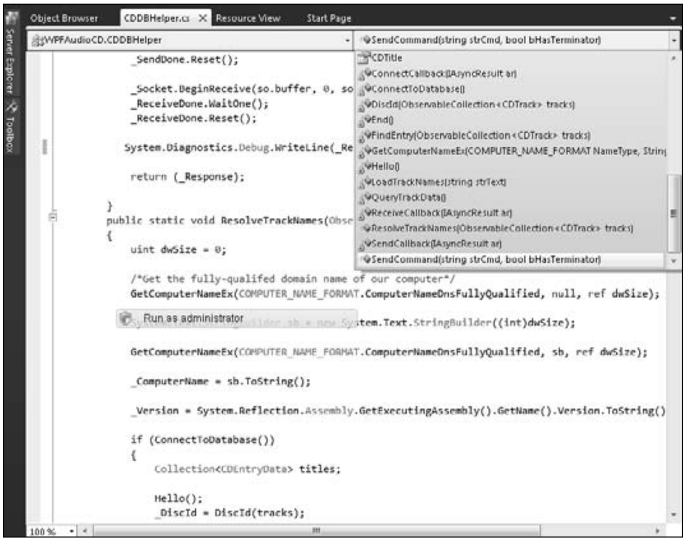
- **Navigation bar:** Allows you to jump to the classes and methods defined in a source code file
- **Bookmarks:** Allow you to place a bookmark in your code

The Navigation bar appears at the top of a source code file in the Code Editor. The left drop-down box lists types or classes that you can find in the source code file. After selecting a type, you can view the type's members in the right drop-down list. Selecting a member takes you to its declaration in source code. Figure 1-9 shows an example of selecting a type's member from the Navigation bar.

A *bookmark* sets a place in code where you can jump to via the Bookmark window. To set a bookmark in source code, do the following:

1. **Position your cursor on the line where you want to set the bookmark.**
2. **Choose View⇨Bookmark Window to open the Bookmark window.**
3. **Click the Toggle button on the far left of the Bookmark window to set the bookmark.**
4. **Type a descriptive name for the bookmark.**

After you set a bookmark, double-click the bookmark in the Bookmark window to jump to that place in the code.


Object Browser
CDDBHelper.cs
Resource View
Start Page
Server Explorer
Toolbox
WPFAudioCD.CDDBHelper
SendCommand(string strCmd, bool bHasTerminator)
CDTitle
ConnectCallback(IAsyncResult ar)
ConnectToDatabase()
DiscId(ObservableCollection<CDTrack> tracks)
End()
FindEntry(ObservableCollection<CDTrack> tracks)
GetComputerNameEx(COMPUTER_NAME_FORMAT NameType, Strin
Hello()
LoadTrackNames(string strText)
QueryTrackData()
ReceiveCallback(IAsyncResult ar)
ResolveTrackNames(ObservableCollection<CDTrack> tracks)
SendCallback(IAsyncResult ar)
SendCommand(string strCmd, bool bHasTerminator)
Run as administrator
100 %

```
            _SendDone.Reset();

            _Socket.BeginReceive(so.buffer, 0, so
            _ReceiveDone.WaitOne();
            _ReceiveDone.Reset();

          System.Diagnostics.Debug.WriteLine(_Re

            return (_Response);

        }
        public static void ResolveTrackNames(Obse
        {
            uint dwSize = 0;

            /*Get the fully-qualifed domain name of our computer*/
            GetComputerNameEx(COMPUTER_NAME_FORMAT.ComputerNameDnsFullyQualified, null, ref dwSize);

                                                 stem.Text.StringBuilder((int)dwSize);

            GetComputerNameEx(COMPUTER_NAME_FORMAT.ComputerNameDnsFullyQualified, sb, ref dwSize);

            _ComputerName = sb.ToString();

            _Version = System.Reflection.Assembly.GetExecutingAssembly().GetName().Version.ToString()

            if (ConnectToDatabase())
            {
                Collection<CDEntryData> titles;

                Hello();
                _DiscId = DiscId(tracks);
```

Figure 1-9: Use the Navigation bar to select the types and members declared in source code.

Visual Studio provides the Object Browser, Class View, and Code Definition for viewing assemblies. See the section on browsing types in Chapter 2 of Book III for more information.

Exploring Visual Basic and C#

Visual Studio supports many programming languages. Visual Basic and C# are two popular .NET programming languages. This section shows you the highlights of using Visual Basic and C# with Visual Studio 2010.

WARNING!

Mastering the basic syntax of programming languages, such as Visual Basic and C#, isn't the same as learning how to program. When you discover how to program, you also discover basic programming constructs, such as conditionals and iterators. By grasping the concepts of programming constructs, you can easily and quickly pick up any language syntax.

Organizing programs

An important element of using a new programming language is figuring out how you should organize your program. You may already know that you use a `Dim` statement to declare variables in Visual Basic and that you use a `using` statement in C# to reference a namespace. That's all well and good, but if you don't know how to organize those statements in your source code file so that they're in the proper order and nested, your program won't compile.

Both Visual Basic and C# have similar approaches to program organization that use these elements:

- **Non-executable statements:** Lines of source code that provide setup and wrapper statements for grouping executable statements, including
  - `Imports` *(Visual Basic) or* `using` *(C#) statements* — `Imports` and `using` statements allow you to use code elements from resources external to your project without fully qualifying the element's namespace each time you use it. These statements create a shortcut, in essence, to the namespace.
  - *Namespace and type declarations*: These statements create wrappers that organize your code.

    Namespace declarations create unique identifiers for the types you declare within the namespace.

    Data type declarations contain executable code. Examples of data types include classes and structures. See Chapter 2 in Book III for more details on types. Visual Basic includes a `Module` statement that allows you to write code without using types.

  Visual Basic uses the `Options` statement to specify how and when syntax checking occurs. `Options` statements are set by default on the project's property pages, so you don't need to include `Options` statements in your source code.

- **Executable statements:** Lines of source code that provide the functionality of your program are *executable statements.* Executable statements are nested inside data type declarations and organized into members. Examples of members include
  - Methods
  - Fields
  - Properties
  - Events

  In C# programming, a *member* is defined as a method, field, property, or event based on the style of programming you use and how you intend to use the member. For example, a class could implement its color as a property member, in which case you would access it as

  ```
  MyClass.Color
  ```

  Implemented as a method, it might look like this:

  ```
  MyClass.GetColor()
  ```

  Coding standards dictate that using the property is preferred over using the method; however, both approaches work and can return the same value.

Visual Basic includes additional executable statements — *procedures* — that can be used as class members or as part of a module. Examples of procedures in Visual Basic include

- `Function`: A set of statements that returns a value
- `Sub`: A set of statements that doesn't return a value
- `Property`: Declares a property with `Get` and `Set` procedures

The long and short of these organizing differences between Visual Basic and C# is that everything must be a data type in C#. In Visual Basic, you don't have to use types to organize your code, but it's recommended.

TIP

The most common data type used is a class. Classes are the fundamental building blocks of object-oriented programming (OOP). You don't have to follow the principles of OOP to use classes in C# or Visual Basic. Some would have you believe that you always write object-oriented code in C#, and that it's easier to write old-style procedural code with Visual Basic. That simply isn't true. Yes, C# forces you to *use* classes and objects, but that's not the same thing as writing object-oriented code. You can write procedural code by using objects. See Chapter 3 in Book III to read more about classes and OOP.

Figure 1-10 shows two functions where you can see the differences between the coding styles in Visual Basic and C#. Notice, for example, that Visual Basic uses `Namespace...End Namespace`, and C# uses `namespace {}`. Note also that because C# is case sensitive, all its keywords are in lowercase (top of Figure 1-10). Comparatively, Visual Basic keywords are in proper case with initial letters capitalized (bottom of Figure 1-10).

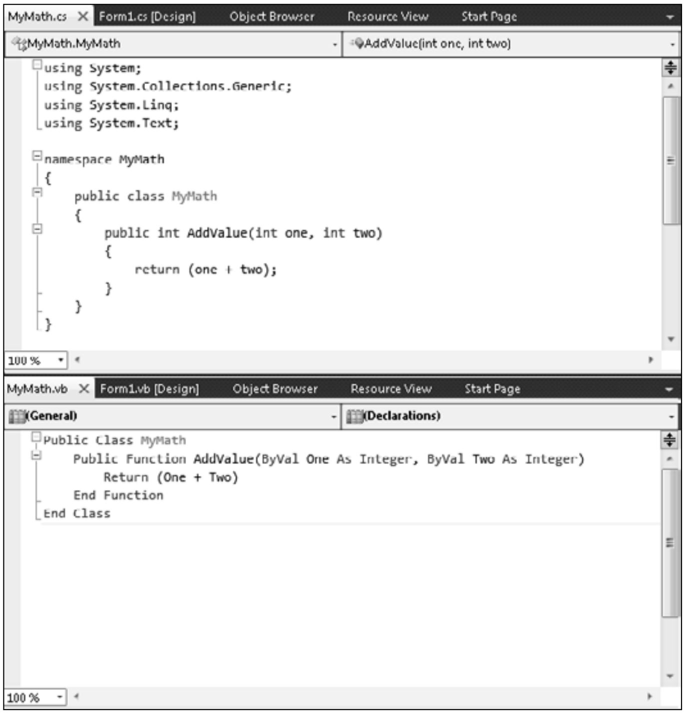
Most development professionals adhere to a set of capitalization standards when it comes to naming variables, classes, and other units of code. The casing standards are consistent, regardless of the programming language used. Chapter 5 in Book III covers these standards.

TIP

Lots of folks say that one or the other language is better, but you can write good code in Visual Basic, and you can write horrible code in C#, and vice versa. What matters is how well you learn how to code, not which language you choose.

Getting started

The following sections provide a few syntax details for C# and Visual Basic. Refer to the Visual Studio documentation for more specifics on programming with either language.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MyMath
{
    public class MyMath
    {
        public int AddValue(int one, int two)
        {
            return (one + two);
        }
    }
}
```

```
Public Class MyMath
    Public Function AddValue(ByVal One As Integer, ByVal Two As Integer)
        Return (One + Two)
    End Function
End Class
```

Figure 1-10: The coding styles in Visual Basic and C# are similar, but their syntax differs slightly.

Beginning at the beginning: The Main procedure

Every executable program requires a procedure called `Main`. The `Main` procedure is the entry point into your application. In Visual Basic Windows applications, the Visual Basic compiler creates a `Main` function for you when the Enable Application Framework check box is selected on the Application tab of the project's property pages.

Only executable programs, such as Windows applications, require a `Main` procedure. Class libraries don't require a `Main` procedure.

The Movie Collection Starter Kit for C# has a good example of a `Main` procedure. In the `Program.cs` source file, you'll find the following `Main` procedure:

```
static void Main()
{
  Application.EnableVisualStyles();
  Application.SetCompatibleTextRenderingDefault(false);
  Application.Run(new MainForm());
 }
```

Declaring variables and assignment

You must declare variables you wish to use and specify their data type. You can assign a value to the variable when you create it or assign a value in a separate assignment statement.

Visual Basic uses the keyword `Dim` to declare variables, like this:

```
Dim variablename as DataType
```

For example, to declare the variable `i` as the data type integer in Visual Basic, you type

```
Dim i As Integer
```

Comparatively, C# doesn't require any special keywords when declaring variables. You declare variables in C# with the following syntax:

```
DataType variablename;
int i;
```

You assign values to variables using the equal (=) operator. For example, to assign the value `7` to the variable `i`, you'd type this:

```
i = 7 "(VB)
i = 7; //(C#)
```

When you read an assignment statement, you say `i` "gets" 7, not `i` "equals" 7. In the case of variable assignment, the equal operator is not the same as equality.

You can declare a variable and assign a value in one statement. To combine declaration and assignment in Visual Basic, you type

```
Dim i As Integer = 7
```

And in C#:

```
int i = 7;
```

See Chapter 2 in Book III for more details on declaring variables and assigning values.

Creating classes and working with objects

The primary means of organizing code in Visual Basic and C# is class declarations. Your executable code is nested within a class declaration. The following code is an example of a class declaration in Visual Basic:

```
Public Class Customer
"executable code here
End Class
```

You use a class using the new keyword, which creates an instance of the class, called an *object.* You use the object to access the executable code inside the class. For example, to create an object called `newCust` from the class `Customer` in Visual Basic, you'd type

```
Dim newCust As New Customer()
```

Here's the equivalent statement in C#:

```
Customer newCust = new Customer();
```

To call an executable procedure on the `newCust` object, you use dot notation, as shown here:

```
newCust.ValidateAddress()
```

You can consume objects in either language, regardless of what language the object is written in. For example, you can consume a Customer object written in Visual Basic using C#, and vice versa.

See Chapter 3 in Book III for more details on declaring classes and creating objects.

Creating executable code

A single line of code is a *statement.* Both Visual Basic and C# provide syntax that allows you to write multiple statements as a block of code.

Examples where you need to use code blocks include `If...Then` statements and `Try...Catch` statements.

C# uses curly braces to encapsulate multiple statements. Visual Basic uses the syntax `Procedure...End Procedure`. Compare these two `If` statements for clarification:

```
"Visual Basic
If flag = True Then
  j = j + 1
  flag = False
End If

//C#
if (flag = true)
 {
    j = j + 1;
    flag = false;
 }
```

Members and procedures

Executable code is implemented as members of a data type, usually a class. Basically, everything inside a class is a member of the class. A procedure that adds two integers and returns a value is a *method.* Variables declared for use within the class are *fields.*

Visual Basic also uses classes and members, but it uses keywords such as `Function` and `Sub` to declare a procedure.

See Chapter 3 in Book III for more details on creating members and procedures in classes.

My Visual Basic

Visual Basic includes an object called `My`, which you can use to access features related to your application and the .NET Framework. For example, you can use `My.Settings` to access your application's configuration settings.

You don't have to declare variables or create an instance to use `My`. You simply type **My** and a dot in the Visual Basic Code Editor, and a list of available objects appears. Three commonly used `My` objects are

- `My.Application`: Sets properties of the current application, such as setting a startup splash screen
- `My.Computer`: Provides access to computer resources, such as the Clipboard and the file system
- `My.User`: Provides information about the user account using the application

Chapter 2: Understanding Data Types

In This Chapter

- **Declaring variables**
- **Seeing the differences between value types and reference types**
- **Understanding how data types are used in the .NET Framework**
- **Handling more than one variable**
- **Finding data types and using them in your code**

The data type you specify for the variable declaration determines everything about how that variable is treated throughout its entire life cycle. This chapter shows the data types available in the .NET Framework and how to create your own data types.

The Rules of Data Types

Data types are the fundamental organizing blocks of code in .NET. Your program can do very little without using data types. If you've never programmed with .NET, you might be thinking of data types in terms of integers and characters. Those are one kind of data type, but other data types are more flexible, sophisticated, and powerful than integers and characters. Examples include classes, structures, and enumerations.

Variables are used to access data that you place in your computer's memory. When you declare a variable, you specify its data type. This is how your computer program knows how to assign meaning to the ones and zeroes that you store in your computer's memory.

From the perspective of the computer, all data is just a set of ones and zeroes. Computer programs apply meaning to those ones and zeroes by using data types. A data type tells your computer how your program can interact with the data and what operations are legal.

Examples of data types are integers and characters. By telling your computer program that a given variable holds an `integer` data type, the computer knows that you can add and subtract the value. The same data type law tells your computer that you can't add and subtract characters. When you use data types, you tell the computer the rules for interacting with your data.

These rules are to your benefit. The compiler for your program uses the rules for your data type to tell you in advance whether your program will break when you run it. Data types allow the compiler and Visual Studio to give you feedback about your code before you even execute it. Giving you feedback early in the process allows you to correct errors before your end users find them.

The many kinds of data types include simple data types (such as integers and characters) as well as complex data types (such as those provided by the .NET Framework). You can create your own data types by combining simple and complex data types.

Making a Declaration

Any time you want to use data in your program, you must first declare a variable to hold the value in memory. When you declare a variable, you do two things:

- **Name your variable.** You create an identifier that allows you to access the variable's value in memory and pass it around in your program.
- **Specify a data type.** You tell the computer how to allocate memory for the variable.

You'll most frequently declare variables in your programs, but you aren't limited to declaring only variables. You can declare constants, enumerations, functions, namespaces, and new data types.

Not all programming languages require you to specify a data type when you declare a variable. Languages (such as the languages of the .NET Framework) that require you to specify a data type are *strongly typed* languages. Specifying a data type at the time of variable declaration

- Enables the Code Editor to use features, such as IntelliSense
- Allows the compiler to provide you feedback if you try to use data types improperly

Declaring a variable is a straightforward process. For example, here's how you declare a variable in Visual Basic and C#:

VB:

```
Dim myVariable as DataType
```

C#:

```
DataType myVariable;
```

After you declare a variable, you might assign a value to it. The first value assigned to a variable is its *initial value.* Assigning a value to a variable for the first time is *initializing* the variable.

You don't have to initialize variables. The .NET Framework provides default initial values for you, depending on the data type; however, the initial values might not be what you expect. Certain kinds of data types are initialized to `null`, for example. You can't work with `null` in your program. You want to make sure that you assign a value to your variables before you start taking actions on those variables in your program.

You can initialize a variable when you declare it. For example, the following statement declares a variable of type `integer` and sets the variable's initial value to `7`.

VB:

```
Dim i As Integer = 7
```

C#:

```
int i = 7;
```

Alternatively, you can assign an initial or subsequent value by using the equal (=) operator, as shown in the following snippets:

VB:

```
i = 6
```

C#:

```
i = 6;
```

Complex data types called *classes* require you to use the new operator when you assign a value. For example, the following code declares a variable of the data type System.Data.DataSet. The second line of code uses the new operator to assign a value to the variable.

VB:

```
Dim ds as System.Data.DataSet
ds = new System.Data.DataSet
```

C#:

```
System.Data.DataSet ds;
ds = new System.Data.DataSet();
```

Using the new operator creates an *instance* of the class. The class is like a template that defines what values can be stored. When you assign an instance of a class to a variable, the variable is like a blank entry form based on the class template.

You can declare and initialize a class in one line:

```
Dim ds as new System.Data.DataSet
```

The .NET Framework's Common Type System

One of the services provided by the .NET Framework is data type management in the form of the Common Type System (CTS). The CTS defines all the rules for how the programming language you use

- Declares types
- Creates new types
- Uses types in your source code

The CTS ensures data type consistency among the programming languages of the .NET Framework.

Understanding the type hierarchy

The CTS has a type hierarchy that provides a base set of data types used by all the .NET programming languages. The root data type for all data types in the .NET Framework is System.Object.

The most common use of data types is declaring variables, as shown in the preceding section. When you initialize a variable, either through assignment or the `new` operator, the computer uses the variable's data type to know how to allocate the memory for the variable. You can use two basic kinds of data types in .NET:

- **Value types:** Simple data types that are built into most programming languages, such as `integer` and `Boolean`.

  The .NET Framework has a set of built-in value types that derive from the data type `System.ValueType`.
- **Reference types:** Complex data types that hold a memory address that points to data stored elsewhere in memory. Examples of reference types include classes, interfaces, strings, arrays, and delegates.

The easiest way to know the difference between value types and reference types is to look at how you declare and initialize variables of either type. Reference types use the `new` operator, whereas value types don't.

For example, the following code declares a variable of the value type `integer` by using the C# keyword `int`:

```
int i = 1;
```

When you create a variable with a reference data type, you use the `new` operator. The code that follows creates a new variable `o` of the data type `System.Object`. Recall that `System.Object` is the root data type in the CTS:

```
System.Object o = new System.Object();
```

Throwing it on the stack or the heap

Value types and reference types are stored differently in memory, which contributes to why they're used differently. Value types are stored on the stack, and reference types are stored on the managed heap. The *stack* — more specifically, the *call stack* — is an area of memory set aside for managing the execution of your program. You can visualize the stack as one memory address stacked on top of another. The *heap,* however, is a large pool of memory where objects that require longer life spans can live. (*Longer life span* means that the object needs to live beyond the execution of a single function on the call stack.)

The stack is more efficient to access than the heap because the stack discards a variable stored in memory as soon as the variable goes out of scope.

Variables stored in the heap, however, are managed by the .NET garbage collector. The `new` operator requests storage to be allocated on the heap for the variable. The garbage collector clears the variable out of memory when it determines that the variable is no longer being used by your program, which might not correspond to the point in time when your program stops using the variable.

By being stored on the stack, value types are directly accessible. Reference type variables, however, return a reference to an address on the heap where the variable's value is actually stored. Figure 2-1 shows an example of value types and reference types in memory. The reference to a reference type is stored on the stack, which is how your program knows how to access the variable.

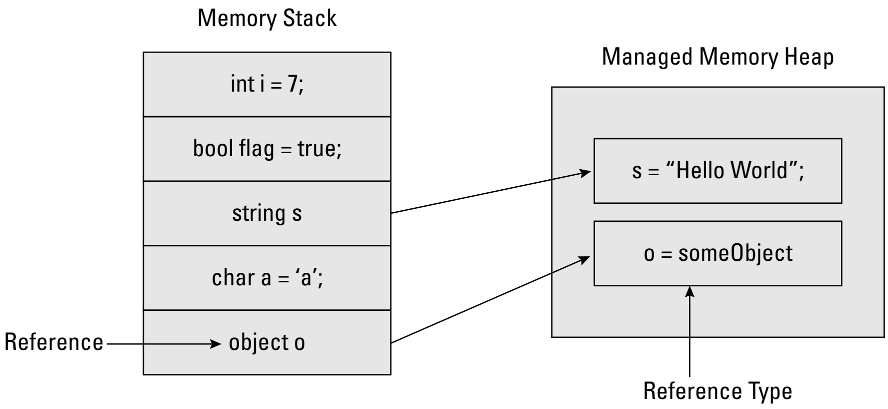


Figure 2-1: Value types are stored on the stack, and reference types are stored on the heap.

Completing your assignments

Another key difference between value types and reference types is how they're handled by assignments. When you assign a variable of a value type, the contents of the value type are copied. When you assign a reference type, only the reference is passed along.

The difference in how value types and reference types are assigned is attributable to how they're stored in memory. A reference type stores only a memory address that points to the actual value. When you copy a reference type, you copy the memory address stored on the stack.

The following code sample uses value types, and the value for variable `i` is copied to the variable `j`:

```
private void TestValueTypes()
{
  int i;
  int j;
  i = 8;
  j = i;
  i = 5;
}
```

When this code executes, j has the value 8, and i has the value 5. j and i are independent variables. The value from i is copied to j.

Now, look at how code using reference types does this. Note that the variables start out pointing to two separate values but wind up pointing to the same value:

```
private void TestReferenceTypes()
{
  System.Data.DataSet ds1 = new DataSet("DataSet 1");
  System.Data.DataSet ds2 = new DataSet("DataSet 2");
  ds2 = ds1;
  ds1.DataSetName = "My DataSet";
}
```

The statement ds2 = ds1; assigns the value referenced in the variable ds1 to the variable ds2. What happens to the value originally referenced by ds2? It's still there, but it can no longer be accessed because ds2 let go of it, as you can see in Figure 2-2. Eventually, the garbage collector recognizes that the object is no longer in use and destroys it.

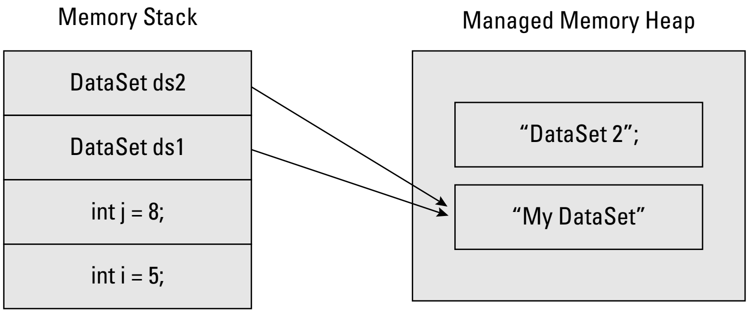


Figure 2-2: Multiple variables can reference the same value.

You may think it's silly that someone would create two variables and then assign one to the other. After you start creating an application, you'd be surprised how easy it is to pass a variable by reference and not understand why your data isn't what you expected.

You need to understand the differences between value types and reference types. Table 2-1 compares value types and reference types.

Table 2-1 Comparison of Value Types and Reference Types

Category	*Value Types*	*Reference Types*
Data accessibility	Directly accessible	Accessed through a reference to the data
Memory allocation	Stack	Managed heap
When is memory freed?	When variable is destroyed	When garbage collection determines no longer in use
Garbage collection	Doesn't use	Uses garbage collection
Initialization	No special initialization required	Must use `new` operator
Default value	Initialized as zero	Initialized as `null`
Null values	Can never be `null`	Throws an exception when you try to use a `null` reference
Assignment	Assignment copies the value	Assignment passes a reference to the value
Base class	Derives from `System.ValueType`	Derives from `System.Object`
Conversion	Can be converted to reference type	Can be converted to value type in some cases

Popular value types

If you've done any programming at all, you're probably familiar with data types, such as `char` and `integer`. These are *value types* in the .NET Framework. The CTS provides a set of built-in value types.

Your programming language provides a set of primitive data types that map to the built-in value type in the CTS. Table 2-2 lists the value types in .NET and their respective keywords in Visual Basic and C#. When you use one of these keywords as a data type in your program, it's compiled as the underlying .NET data type listed.

A *primitive data type* is a type that the programming language's compiler natively understands.

Table 2-2 Built-In Value Types and Their Language Keywords

.NET Data Type	*Visual Basic Keyword*	*C# Keyword*	*Description*
`System.Boolean`	`Boolean`	`bool`	True or false
`System.Byte`	`Byte`	`byte`	Unsigned integer with values ranging from 0 to 255
`System.Char`	`Char`	`char`	Represents characters in the Unicode Standard, such as the letter *a*
`System.Decimal`	`Decimal`	`decimal`	Decimal numbers ranging from –79,228,162,514,264,337,593,543,950,335 to +79,228,162,514,264,337,593,543,950,335
`System.Double`	`Double`	`double`	Fifteen decimal points of precision for binary floating-point arithmetic
`System.Int16`	`Short`	`short`	Signed integer with values ranging from –32,768 to +32,767
`System.Int32`	`Integer`	`int`	Signed integer with values ranging from –2,147,483,648 through +2,147,483,647
`System.Int64`	`Long`	`long`	Signed integer with values ranging from –9,223,372,036,854,775,808 through +9,223,372,036,854,775,807
`System.Sbyte`	`Sbyte`	`sbyte`	Signed integer with values ranging from –127 to +127
`System.Single`	`Single`	`float`	Seven decimal points of precision for binary floating-point arithmetic
`System.UInt16`	`Ushort`	`ushort`	Unsigned integer values ranging from 0 to 65,535
`System.UInt32`	`UInteger`	`uint`	Unsigned integer values ranging from 0 to 4,294,967,295
`System.UInt64`	`Ulong`	`ulong`	Unsigned integer values ranging from 0 to 18,446,744,073,709,551,615

You might notice that many data types represent numbers. The amount of memory used by each of these data types corresponds to the range of values that the data type can store.

Use the `System.Decimal` data type for financial calculations where rounding errors can't be tolerated.

Most of the .NET programming languages have a keyword for a data type that represents strings. Strings aren't value types: They're reference types. See the next section, "Popular reference types," to read more about strings.

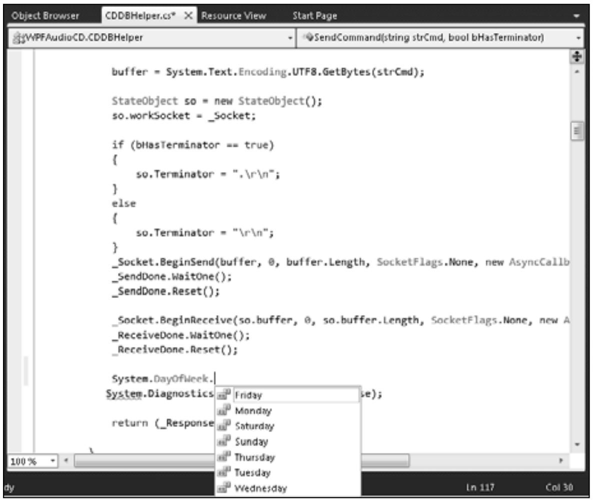
In addition to the value types listed previously, the .NET Framework includes two other programming elements that define value types:

- **Structures:** A *structure* is a data type comprising other data types. You use a structure to consolidate other data types into a single, named data type. Examples of two commonly used structures provided by the .NET Framework are `System.DateTime` and `System.Guid`.

  The Visual Basic keyword that corresponds to `System.DateTime` is `Date`. C# doesn't have a keyword. `guid` (rhymes with *squid*) is short for *globally unique identifier.* You use a `guid` any time you need a unique identifier.

  You use structures to create your own data types. See the section "Creating Your Own Types" for examples of using structures.

- **Enumerations**: An *enumeration* defines a set of constants that can be accessed by the name applied. `System.DayOfWeek` is an example of an enumeration that enumerates the days of the week, as shown in Figure 2-3. Enumerations derive from `System.Enum`.



```
            buffer = System.Text.Encoding.UTF8.GetBytes(strCmd);

            StateObject so = new StateObject();
            so.workSocket = _Socket;

            if (bHasTerminator == true)
            {
                so.Terminator = ".\r\n";
            }
            else
            {
                so.Terminator = "\r\n";
            }
            _Socket.BeginSend(buffer, 0, buffer.Length, SocketFlags.None, new AsyncCallb
            _SendDone.WaitOne();
            _SendDone.Reset();

            _Socket.BeginReceive(so.buffer, 0, so.buffer.Length, SocketFlags.None, new A
            _ReceiveDone.WaitOne();
            _ReceiveDone.Reset();

            System.DayOfWeek.
           System.Diagnostics                        se);

            return (_Response
```

Figure 2-3: Enumerations define a set of constants.

Popular reference types

All the classes in the .NET Framework are reference types, so-called because the variable holds a reference to the value — not the actual value itself.

All classes in the CTS derive from `System.Object`. You could say that makes `System.Object` the most popular reference type. Hopefully, however, you aren't actually declaring your variables by using the `System.Object` type. Although doing so isn't incorrect, try to use the most specific data type possible.

C# provides the `keyword` object, and Visual Basic uses `Object` to map to the `System.Object` type in CTS.

One of the most popular reference types is `System.String`. Many people think that `System.String` is a value type, but it's actually an array of characters. That's probably because you don't have to use the `new` operator when you create a new string. You declare and use a string variable similarly to how you use value types, such as `integer` and `Boolean`.

Visual Basic and C# both provide a keyword to represent `System.String`. The following code declares and initializes a string variable:

VB:

```
Dim s As String = "Hello"
```

C#:

```
string s = "Hello";
```

Another popular reference type is `System.Exception`. Any time your program throws an exception, the exception you see is a reference type derived from `System.Exception`.

See the section "Browsing Types," later in this chapter, to see how you can find more reference types in the .NET Framework.

Creating Your Own Types

Developers long ago figured out that describing a business domain strictly in terms of `int`s and `char`s isn't easy. You can't bend the business to fit the limitations of a programming language. No, instead, you must use a programming language that allows you to model whatever real-world problem you're trying to solve.

The .NET Framework allows you to create your own data types. You can use your own data types to

- ✦ Model your business in your software.
- ✦ Provide utility features.
- ✦ Customize existing types that don't quite fit your needs.

You can use the following kinds of data types to create your own data types:

- ✦ **Classes:** *Classes* are reference types that derive from `System.Object`. Class types define the data and behavior of a variable. In other words, classes define the data that a variable can store and provide procedures that act on that data. For example, a `Customer` class may store a customer's name and unique identifier. It may include an `AccountBalance()` procedure that returns the customer's current balance.

  Visual Basic and C# provide the `Class` statement and `class` keyword, respectively, for creating your own class data types. See the next chapter for more information on creating classes.

- ✦ **Structures:** *Structures* are value types that derive from `System.ValueType`. Structures can store virtually all the same data and behaviors as a class.

  You use the `Structure` statement in Visual Basic to create a structure. C# provides the `struct` keyword:

  - *To create a Customer structure in Visual Basic, type the following code:*

```
Structure Customer
  Dim m_firstName As String
  Dim m_lastName As String
  ReadOnly Property Name() As String
    Get
     Return m_firstName + " " + m_lastName
    End Get
  End Property
  WriteOnly Property FirstName()
    Set(ByVal value)
      m_firstName = value
    End Set
  End Property
  WriteOnly Property LastName()
    Set(ByVal value)
      m_lastName = value
    End Set
  End Property
End Structure
```

- *To create a variable by using the Customer data type, type the following code:*

  ```
  Dim cust As New Customer
  ```

- *To assign values to a variable, type the following code:*

  ```
  cust.FirstName = "John"
  cust.LastName = "Smith"
  ```

The following line of code assigns the variable's `Name` property to a text box:

```
txtCustomerName.Text = cust.Name
```

Notice that the `cust` variable accesses only the structure's properties, and not the variables declared at the beginning of the structure. The structure's properties, in turn, access the variable declared at the top of the structure. The variables and properties of the structure are the *members* of the structure. See the next chapter to read more about a data type's members.

✦ **Enumerations:** *Enumerations* are value types that derive from `System.ValueType`. You define a set of constants, such as the days of the week, in an enumeration. You can use the `System.Enum` data type to access additional features of enumerations.

Visual Basic provides the `Enum` statement, and C# uses the `enum` keyword for declaring enumerations. The following code shows an enumeration in C#:

```
enum Fiber
{
  Angora,
  Mohair,
  Wool
}
```

With object-oriented programming (OOP) techniques, you can extend virtually all the types provided in the .NET Framework to meet your specific needs. See Chapter 3 in Book III for more information about OOP.

You often create user-defined types in a class library project. You can reference the class library in a Windows project or Web application when you need to use your user-defined types.

You can use your class and structure data types just like you do any other data type. You can declare them as variables, pass them as parameters, and return them from procedures. Use enumerations any time you need to reuse a set of values throughout your application.

At first blush, you may think there isn't much difference between structures and classes. Recall, however, that structures are value types and classes are reference types. As a result, structures often use less memory than classes. Each time a value type like a structure is passed around in a program, a copy of the structure is made. So, what starts out using less memory could end up consuming quite a bit. For this reason, you'll often find that classes are used more than structures. Even though classes are initially more expensive to create because there is both memory allocation and startup code to execute, their memory usage is often more economical throughout their lifetime.

In general, you should create structures when the data type you're creating is small in size, like an integer, and you expect it to be short-lived.

When There's More than One

Quite often, you need to handle more than just one of something. Rarely does your business have just one customer or one product. You're usually dealing with sets of things. The .NET Framework provides many data types for dealing with situations when you have more than one item.

TIP

Data types that can handle sets of data are often referred to as *data structures* or *collections.*

The collection-related data types provided by the .NET Framework often allow you to

- Add, remove, and modify individual elements.
- Copy elements to another collection.
- Sort and index elements.
- Iterate through a set of elements.

The .NET Framework provides several data types you can use to manage collections. The two biggies are

- `Array` **class:** An *array* is a set of data of all the same data type. You set the size of the array when you declare it. Arrays have been the staple data structure for a long time.

  TIP

  Picture an array as an Excel spreadsheet. A one-dimensional array is like a single row in the spreadsheet. A multidimensional array has more than one row.
- `System.Collections` **namespace:** Other kinds of collections, such as lists and hashtables, are found in the `System.Collections` namespace.

A namespace references a set of data types. A namespace isn't a data type itself. No `System.Collections` data type exists in the .NET Framework. Rather, you use the `System.Collections` namespace to access data types used to manage collections.

Another kind of data structure provided by the .NET Framework is the ADO. NET, which is an in-memory representation of a database with tables, columns, and rows. For more on `DataSet`s, turn to Book V, Chapter 3.

Using arrays

The .NET Framework provides the `System.Array` class for creating arrays. An *array* defines a set of data that all have the same data type. You can define an array to use any kind of data type, such as a set of integers or a set of strings. You can even define an array by using your own custom data types.

All items in an array must be of the same data type.

An array has the following properties:

- **Elements:** Each item that you add to an array is an element of the array. The data type of the element is the *element type.*
- **Index:** The *index* is the position of each element in the array. Arrays use zero-based indexes, so the first value in an array has an index of zero.
- **Length:** This is the total number of elements in the array.
- **Rank:** This is the number of dimensions in the array. A one-dimensional array has one row of data. A two-dimensional array has multiple rows.
- **Bounds:** These are the lower and upper bounds of an array that define the starting and ending index for an array's elements. For example, an array with four elements has a lower bound of zero and an upper bound of three.

Declaring arrays is similar to declaring other types of data. In Visual Basic, you append parentheses to the variable's identifier:

```
Dim dailyWeights() As Integer
```

To declare a multidimensional array in Visual Basic, you place a comma inside the parentheses for each additional dimension, such as

```
Dim dailyWeights(,) As Integer
```

In C#, you append brackets to the element's data type when you declare an array:

```
int[] dailyWeights;
```

Similar to Visual Basic, you use commas to create multidimensional arrays:

```
int[,] dailyWeights;
```

You can also create arrays of arrays, which are called *jagged arrays.* You add extra sets of parentheses or brackets for each nested array. A declaration for a jagged array in Visual Basic looks like this:

```
Dim dailyWeights()() As Integer.
```

Declaring an array doesn't actually create the array. Because arrays are reference types, you use the `new` operator to create the array and assign it to the variable you declare. For example, to create an array with five elements and assign it to the one-dimensional dailyWeights array using Visual Basic, you'd type

```
dailyWeights = New Integer(4) {}
```

Recall that arrays have a zero-based index. Inside the parentheses, you place the array's upper bound, which is `4` in this example. The array's length is `5` because you start counting at zero.

Use the curly braces to place values into the array, as the following Visual Basic code shows:

```
dailyWeights = New Integer(4) { 155, 153, 154, 152, 150 }
```

Here's the equivalent statement in C#:

```
dailyWeights = new int[5] { 155, 153, 154, 152, 150 };
```

You may have noticed some subtle differences in syntax between Visual Basic and C#. Most notably, in the Visual Basic statement, you use the upper bound; in C#, you use the array's length. If you switch back and forth a lot between the two languages, maybe you can get a tattoo so that you can keep it straight.

You don't need to size an array when you're initializing the array in the same statement. Supplying five values automatically creates an array of `length = 5`. For example, the following statements are equivalent:

```
dailyWeights = new int[5] { 155, 153, 154, 152, 150 };
dailyWeights = new int[] { 155, 153, 154, 152, 150 };
```

Here are three steps to using arrays:

1. **Declare the array variable.**
2. **Create the array.**
3. **Initialize the array with values.**

You can perform each step discretely or combine all three steps into one statement, as shown in the following C# statement:

```
int[] dailyWeights = new int[5] { 155, 153, 154, 152, 150 };
```

The equivalent statement in Visual Basic is

```
Dim dailyWeights() As Integer = New Integer(4) {155, 153, 154, 152, 150}
```

You supply three pieces of information to declare and create an array:

- The element's data type
- Its rank
- The upper bound or the size of the array

To access the elements in an array, use an indexer to specify the position of the element you wish to access. For example, the following C# code accesses the third element in an array of integers:

```
dailyWeights[2] = 175;
```

REMEMBER

Arrays use a zero-based index, which means that you start counting from zero.

Using System.Collections

The .NET Framework provides many kinds of collections that you can use when you need to handle more than one of something at a time. Table 2-3 lists the specialized collection types you can find in the `System.Collections` namespace.

You can group collection data types in the `System.Collections` namespace based on the mechanism used to access elements in the collection:

- **Indexed:** These access elements by using their position in the list of elements.
- **Keyed:** These access elements by using the key in a key/value pair.
- **Neither indexed nor keyed:** Data types provide access methods other than an indexer or a key.

Collections that use indexes are *lists.* Keyed collections are *dictionaries.*

Table 2-3 Data Types in the System.Collections Namespace

Accessor	*Collection Type*	*Data Type*	*Description*	*Example*
Both	Dictionary	`SortedList`	A set of key/value pairs sorted by the key	A glossary of terms
Indexed	Collection	`BitArray`	An array of Boolean values	Whether your dog responds to the Come command given successively at the bark park
Indexed	List	`ArrayList`	An array of variable size	The number of debits to your bank account in the next 30 days
Keyed	Dictionary	`Hashtable`	A set of key/value pairs sorted by the key's hash number	The movies being shown at a theater
Neither	Collection	`Stack`	Last-in, first-out (LIFO) list	A pile of football players on a quarterback
Neither	Collection	`Queue`	First-in, first-out (FIFO) list	Line at the grocery store

The `System.Collections` namespace defines the `DictionaryEntry` structure, which represents a key/value pair in a dictionary collection type. See the upcoming section "Iterating through arrays and collections" to see a code sample that uses the `DictionaryEntry` structure.

The collection data types found in the System.Collection namespace are all classes. You use the new operator to create a new instance of a collection. The following code shows a Hashtable:

```
Dim ht As New Hashtable
```

Important actions you take on collections, such as a Hashtable, include adding and removing elements. A Hashtable is a key-based collection, so you add elements by using a key/value pair. The following code sample shows how to add elements to a Hashtable:

```
ht.Add("Screen 1", "Shane")
ht.Add("Screen 2", "My Friend Flicka")
```

Use the Remove method to remove an element from the hashtable:

```
ht.Remove("Screen 1")
```

You supply the key when you want to remove the element.

Of course, you aren't limited to using just primitive types and strings in your collections. You can use any data type, including your user-defined data types. For example, instead of placing a movie title in the hashtable, you could place a value created from a Movie class. The Movie class might store a movie's title, actors, release date, and show times. See the earlier section on creating your own types for more information.

The .NET Framework provides two other namespaces for using collection data types:

- System.Collections.Generic: Provides data types that allow you to create type-specific collections. In a type-specific or strongly typed collection, you specify in advance the type of data that can be placed into the collection.

  Type-specific collections are *generic collections.* All the collections listed in Table 2-3 have generic counterparts.

  For example, the Dictionary data type is the generic version of the Hashtable data type. To create a keyed collection that accepts only the Customer data type, you use the Dictionary data type, as shown here:

  ```
  Dim invoiceCustomers As New Dictionary(Of String, Customer)
  ```

  You add objects of the type Customer with a string key that represents their customer ID by using the Add method, as shown in the following code:

  ```
  invoiceCustomers.Add("1234", cust)
  ```

- System.Collections.Specialized: Contains a set of strongly typed or specialized collections. For example, StringDictionary is a generic hashtable that works only with the data type string. If you try to put some other data type (such as an integer) into a StringDictionary, you receive an exception.

Collections are used extensively throughout the .NET Framework. For example, Windows Forms and Web Forms have a set of control collections. You can iterate through a form's collection of controls to add controls or find a specific control you wish to use.

Iterating through arrays and collections

An important task when working with collections is to be able to step through the elements within the collection. C# and Visual Basic provide the foreach and For Each statements, respectively, for iterating through a collection.

For example, the following code shows iterating through an ArrayList using For Each in Visual Basic:

```
Dim list As New ArrayList
list.Add(1)
list.Add(2)
list.Add(3)
For Each i as Integer In list
  Dim j As Integer
  j = j + i
Next
```

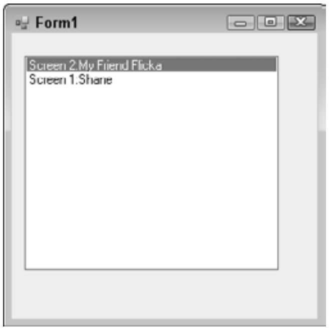
The For Each statement in the preceding sample executes once for each element in the ArrayList for a total of three times. On third execution, it's j = 6 because 1 + 2 + 3 = 6.

The variable j is in scope only while execution is inside the For Each statement. After execution moves off the Next statement the third time, you can no longer access j. If you want to use the variable j outside the For Each statement, you need to declare it outside the For Each statement.

Keyed collections use the DictionaryEntry structure for iterating through values, as the following C# code sample shows:

```
Hashtable ht = new Hashtable();
ht.Add("Screen 1", "Shane");
ht.Add("Screen 2", "My Friend Flicka");
foreach (DictionaryEntry de in ht)
{
   lstMovies.Items.Add(de.Key + ":" + de.Value);
}
```

Figure 2-4 shows an example of the key/value pair from the hashtable in a list box.



```
Screen 2:My Friend Flicka
Screen 1:Shane
```

Figure 2-4: Iterate through the hashtable to display key/value pairs in a list box.

Collections in the real world

You have many choices for working with collections. Here are some guidelines to help you decide:

- **Choose arrays over collections** when the number of elements is known and not expected to grow.
- **Choose collections over arrays** any time you find yourself looking for methods, such as `Add`, `Remove`, `Item`, or `Count`.
- **Choose generic collections** when you know in advance the data type you want to store in the collection. Generic collections perform better than nongeneric collections because their code doesn't have special cases to consider.

Besides selecting the best type of collection for the job, you must also consider how you'll actually use the collection in your code. If you're creating your own data types, you need to consider how to manage multiples of your data types. You have a few options:

- Extend the collections classes provided by .NET.
- Wrap an existing collection by creating a class to manage it.

In most cases, you'll probably wrap an existing collection when you want to provide your own custom collections. You have two approaches to wrapping an existing collection:

- **Create a new data type that wraps the collection.**

  For example, you can create a data type called `MovieCollection` that allows you to handle a collection of movies without thinking about the underlying collection actually used.

- **Use a collection in an existing data type.** An alternative approach to creating a separate wrapper collection is to include the collection in your data type. For example, you could create a generic collection of `Movie` types that you access from within your `Movie` type.

Converting Types

An important task when working with variables is converting a variable of one data type to another data type. For example, a variable with the Boolean value `True` isn't the same as a variable with the string value `true`. They may look the same, but the language sees the Boolean variable as either on or off, and the string variable as an array of characters.

A variable's data type determines how much memory is allocated for the variable. When you convert from one data type to another, you're essentially asking the computer to give you more or less memory. Here are the two kinds of conversions that you can perform:

- **Widening:** Going from a smaller data type to a larger data type
- **Narrowing:** Going from a larger data type to a smaller data type

You risk data loss with narrowing conversions.

The syntax that you use to widen or narrow depends on whether the conversion is

- **Implicit:** Implicit conversions don't require any special syntax in order for the conversion to occur. For example, the following code implicitly converts a `Boolean` value to a `string` value:

  ```
  Dim b As Boolean = True
  Dim s As String = b
  ```

- **Explicit:** Any time you have to use special syntax to convert a variable, you make an *explicit conversion.* Explicit conversions are often necessary when you perform a narrowing conversion.

The .NET Framework provides the `System.Convert` class, which you can use to explicitly convert from one type to another. For example, the following code converts a `string` to an `integer` in C#:

```
string s = "1000";
int i = System.Convert.ToInt32(s);
```

C# provides the `cast` operator for performing explicit conversions. The following code converts from an `integer` value to a `byte`, which is a narrowing conversion:

```
int i = 255;
byte b = (byte)i;
```

Narrowing conversions can cause loss of data. For example, take a look at the following code sample:

```
int i = int.MaxValue;
byte b = (byte)i;
```

The maximum value of an `integer` data type is north of two million. So what's the value in the variable `b` after the conversion of integer `i`? It's `255`. A byte holds values from only 0 to 255.

Visual Basic provides the `CType` function, which you can use for explicit conversions. You pass an expression to convert and the data type to convert to the `CType` function, as shown here:

```
Dim s As String = "255"
Dim b As Byte = CType(s, Byte)
```

Visual Basic has type conversion functions for each primitive data type and a function each for converting reference types and strings. For example, the following code is equivalent to using `CType` in the preceding sample:

```
Dim s As String = "255"
Dim b As Byte = CByte(s)
```

Any implicit conversion can be explicitly stated. The following code explicitly converts a `Boolean` value to a `string` value:

```
Dim b As Boolean = True
Dim s As String = Convert.ToString(b)
```

There's no harm in using an explicit conversion in place of an implicit conversion. You should use explicit conversion any time you want to make it clear to readers of your code that a conversion is occurring.

You aren't restricted to converting between primitive types and strings. You can convert any data type in the .NET Framework, your programming language, or your user-defined data types. `System.Convert`, `CType`, and the `cast` operator all allow any kind of data type for making conversions. The catch, of course, is that the data types you're converting must be compatible.

In order for a conversion to be successful, a conversion operator must be defined for the type you want to go from to. See the Visual Studio documentation for a list of available conversions for your language.

Converting value types (such as integers and Booleans) to reference types (such as strings and objects) is *boxing.* Boxing and its converse — *unboxing* — occur any time you use a value type when a reference type is expected. Passing value types as parameters when a reference type is expected is one example of when a value type is boxed.

All data types derive from `System.Object`; therefore, you can convert a variable of any data type to `System.Object`.

You can use the `System.Type` class to find out more information about a data type, such as the data type from which the data type is derived, as well as whether the data type is a value type or a reference type.

Meet the Nullable Types

Often, when working with values from databases, you encounter `null` values, which are undefined values. A `null` value can make your program blow up when the program is expecting to see an `integer` or a `Boolean` or a `string`. To help you process and anticipate `null` values, the .NET Framework includes a data type called `System.Nullable`.

You use `System.Nullable` to tell your program to accept a `null` value in your variable. `System.Nullable` provides the following properties:

- `HasValue`: Returns a `true` or `false` value indicating whether the variable has a value or is `null`.
- `Value`: Retrieves the variable's value. You use the `HasValue` property to test that the variable contains a value before using the `Value` property.

`System.Nullable` works with value types. *Values types* are primitive data types, such as `integer` and `char`. By definition, value types can't store `null` values. Reference types, such as strings, can store `null` values. As a result, it's not necessary for `System.Nullable` to work with reference types.

WARNING!

That’s not to say that null reference types can’t wreak the same kind of havoc in your program as trying to assign a `null` value to a value type. You should test your reference types for `null` values before you try to access the value.

When you declare a nullable value type, you tell `System.Nullable` which value type you wish to use. The following Visual Basic code sample creates a nullable `integer`:

```
Dim i As System.Nullable(Of Integer)
```

The equivalent declaration in C# is

```
System.Nullable<int> i;
```

C# provides the question mark (`?`) shortcut operator you can use when declaring nullables. The following statement is equivalent to the preceding statement:

```
int? i;
```

By declaring a variable as nullable, you can use the `HasValue` property to test for a `null` value. In the following code sample, if a nullable integer `i` has a value, the value is returned. Otherwise, the procedure returns `0` (zero).

```
int checkValue(int? i)
{
  if (i.HasValue == true)
     return i.Value;
  else
     return 0;
}
```

Browsing Types

The .NET Framework has hundreds of data types. Your own code base may have dozens — possibly even hundreds — of data types. Visual Studio provides Object Browser for perusing the vast libraries of data types available to you.

You use Object Browser any time you need to

- ✦ Find a data type.
- ✦ View the members of a data type, such as properties and methods.
- ✦ View a description and get help for a data type.

Book III
Chapter 2

Understanding Data Types

You open Object Browser using the View menu or the key combination Ctrl+Alt+J. You don't need to have a project open to use Object Browser. Your open projects appear in Object Browser.

Setting the scope

You'd be quickly overwhelmed if you had to look at all the data types in Object Browser at once. Instead, Object Browser allows you to limit the scope of the types you view at any one time to the following:

- .NET Framework
- Third-party components
- Your own projects and components

To view only the components in the .NET Framework, follow these steps:

1. **Press Ctrl+Alt+J to open Object Browser.**
2. **Click the Browse drop-down list on the Object Browser toolbar.**

   A list of browsing scopes appears.

   Object Browser displays data types from these browsing scopes:

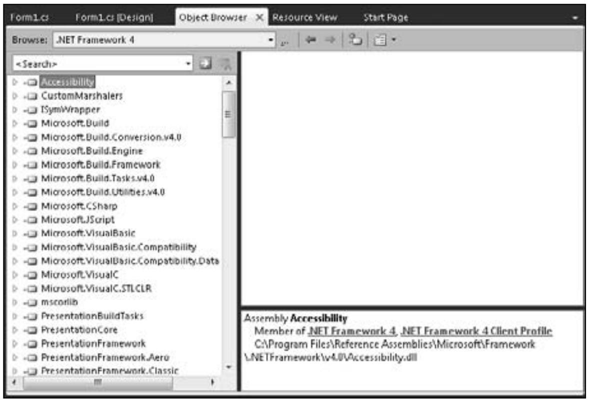
   - *All Components*: Displays the data types from the other options
   - *.NET Framework*: Displays data types found in the .NET Framework
   - *My Solution*: Displays data types created and referenced in the open solution
   - *Custom Component Set*: Displays data types from a third-party component.
3. **Select .NET Framework 4 from the drop-down list.**

   The assemblies in the .NET Framework appear in the Objects pane on the left, as shown in Figure 2-5.

TIP

Use the My Solution browsing scope to view the assemblies referenced by your project.

Alternatively, you can use the Object Browser Search drop-down list to search for a word. Search is limited to the browsing scope selected in the Browse drop-down list. See Chapter 7 of Book III for an example of using Search in Object Browser.



Browse: .NET Framework 4

Accessibility
CustomMarshalers
ISymWrapper
Microsoft.Build
Microsoft.Build.Conversion.v4.0
Microsoft.Build.Engine
Microsoft.Build.Framework
Microsoft.Build.Tasks.v4.0
Microsoft.Build.Utilities.v4.0
Microsoft.CSharp
Microsoft.JScript
Microsoft.VisualBasic
Microsoft.VisualBasic.Compatibility
Microsoft.VisualBasic.Compatibility.Data
Microsoft.VisualC
Microsoft.VisualC.STLCLR
mscorlib
PresentationBuildTasks
PresentationCore
PresentationFramework
PresentationFramework.Aero
PresentationFramework.Classic

Assembly **Accessibility**
Member of .NET Framework 4, .NET Framework 4 Client Profile
C:\Program Files\Reference Assemblies\Microsoft\Framework
\.NETFramework\v4.0\Accessibility.dll

Figure 2-5: The assemblies from the selected browsing scope appear in the Objects pane.

Setting the view

Object Browser displays data types of all kinds, including classes, enumerations, and interfaces. By default, data types appear in assembly containers; however, there are many different views, such as

- **Assemblies:** The physical files in which the data type is defined. *Assemblies* are the DLL files that are output when your source code is built.
- **Namespaces:** The logical namespace in which the data type is defined
- **Object types:** The kind of data type, such as class, enumerator, or structure

To view the data types by namespaces

1. **Right-click inside the Objects pane.**

   A shortcut menu appears.

2. **Choose View Namespaces.**

   The data types are grouped by their namespaces.

   To group the data types by assembly, repeat Step 1 and choose View Containers from the shortcut menu.

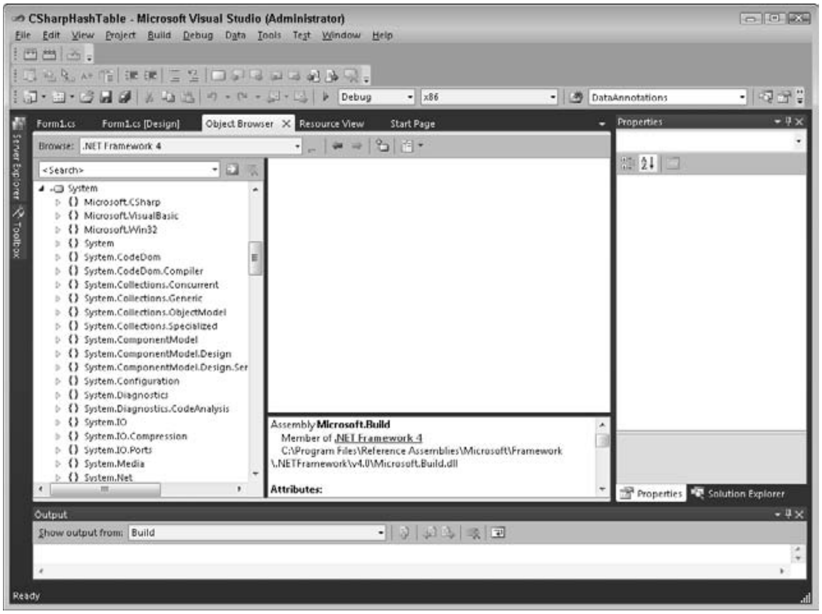
   Choose Group by Object Type from the shortcut menu to group the data types by the type of data type.

To view the physical assembly file where data is defined

1. **Group the data types by assembly, as described in the preceding set of steps.**

2. **Click the assembly you wish to view.**

   The assembly's name, path, and attributes appear in the Description pane. Figure 2-6 shows the assembly information for the `System` assembly.



System
Microsoft.CSharp
Microsoft.VisualBasic
Microsoft.Win32
System
System.CodeDom
System.CodeDom.Compiler
System.Collections.Concurrent
System.Collections.Generic
System.Collections.ObjectModel
System.Collections.Specialized
System.ComponentModel
System.ComponentModel.Design
System.ComponentModel.Design.Ser
System.Configuration
System.Diagnostics
System.Diagnostics.CodeAnalysis
System.IO
System.IO.Compression
System.IO.Ports
System.Media
System.Net

Assembly **Microsoft.Build**
Member of .NET Framework 4
C:\Program Files\Reference Assemblies\Microsoft\Framework\.NETFramework\v4.0\Microsoft.Build.dll

Attributes:

Figure 2-6: Click an assembly to view information about the assembly.

Viewing data types

You can use Object Browser to view all kinds of information about data types, including the following data types:

- Assembly and namespace
- Members, such as properties and methods
- Base data type and derived data types

Here's how to view a data type in Object Browser:

1. **Set your browsing scope and view.**
2. **Click the plus sign (+) next to the container of data types.**

   Depending on how you set up your view, you might also have to expand the Namespaces and Object Type folders to access the actual data types.
3. **Click the data type to view its members and description.**

For example, to access the `System.Enum` data type with the data types grouped by assemblies and object types, follow these steps:

1. **Click the plus sign (+) next to the `mscorlib` assembly.**

   The Namespaces folder appears.

   The `mscorlib` assembly contains the core namespaces of the .NET Framework.

2. **Click the plus sign (+) next to the Namespaces folder.**

   A list of namespaces found in the `mscorlib` assembly appears.

3. **Click the plus sign (+) next to the System Namespace folder.**

   A list of Object Type folders appears.

   Note that the .NET Framework has a `System` assembly and a `System` namespace. The `System` namespace spans across both the `mscorlib` and `System` assemblies.

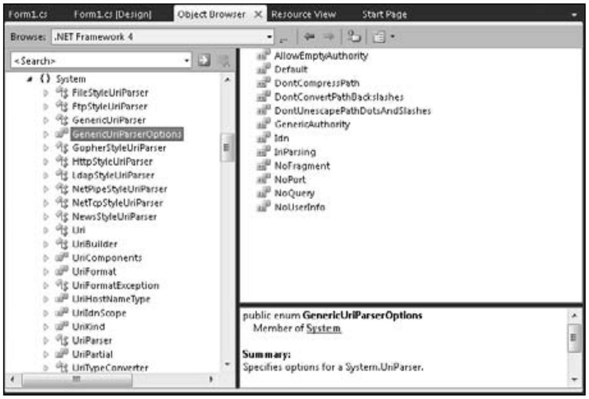
4. **Click the Structures folder.**

   A list of structure data types appears.

5. **Click the `Enum` data type.**

   The members and description appear in the browser.

   To view a member's description, click the member, as shown in Figure 2-7.



Browse: .NET Framework 4

System
FileStyleUriParser
FtpStyleUriParser
GenericUriParser
GenericUriParserOptions
GopherStyleUriParser
HttpStyleUriParser
LdapStyleUriParser
NetPipeStyleUriParser
NetTcpStyleUriParser
NewsStyleUriParser
Uri
UriBuilder
UriComponents
UriFormat
UriFormatException
UriHostNameType
UriIdnScope
UriKind
UriParser
UriPartial
UriTypeConverter

AllowEmptyAuthority
Default
DontCompressPath
DontConvertPathBackslashes
DontUnescapePathDotsAndSlashes
GenericAuthority
Idn
IriParsing
NoFragment
NoPort
NoQuery
NoUserInfo

public enum **GenericUriParserOptions**
Member of System

Summary:
Specifies options for a System.UriParser.

Figure 2-7: Click a data type's member to view its description.

A data type's base data type can give you clues about whether the type is a value type or reference type. Viewing the type's derived data types shows you more specific implementations of the data type that might be more appropriate for you to use.

You can view a data type's base type and any derived types by expanding the data type. In the case of the `System.Enum` data type, its base type is `System.ValueType`. Many types are derived from `System.Enum`, as shown in Figure 2-8.

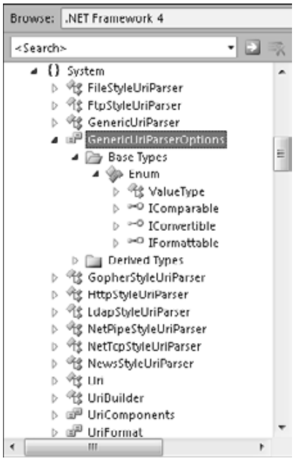


Figure 2-8: Expand a data type to view its base types and derived types.

You may have noticed that the data types, members, and other items in Object Browser have icons next to them. A different icon is used to represent each kind of data type, such as classes or structures. Search for the topic "Class View and Object Browser Icons" in the Visual Studio Help documentation for a summary of the icons used.

Viewing source code

Visual Studio provides several tools for browsing and navigating source code, including

- **Class View:** Use Class View to display a hierarchical view of the solution you're developing.
- **Code Definition:** This displays a read-only view of the source code for the selected object.

You use the Code Definition window in conjunction with the Class View or Object Browser to view an object's source code. When viewing objects from outside your project, the Code Definition window displays only the source code's type and method declarations and comments. You can't actually view the source code that implements the object.

The Code Definition window doesn't work in Visual Basic.

To use the Code Definition window with Class View, follow these steps:

1. **Open a C# project in Visual Studio.**

   If you don't have an existing C# project, you can open one of the C# Starter Kits, such as the Movie Collection Starter Kit, from the New Project window.

2. **Press Ctrl+Shift+C to open Class View.**

   The project appears in the Class View window.

3. **Choose View➪Code Definition Window.**

   The Code Definition window appears.

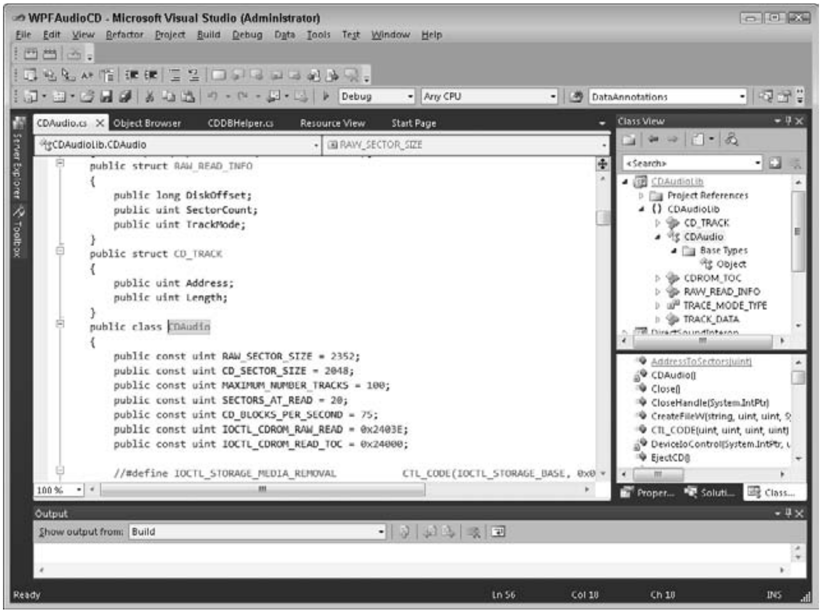
4. **Expand the classes in Class View and click one of the objects.**

   The object's methods appear in the bottom pane of Class View.

   Class View and Object Browser use a number of icons to represent different kinds of objects. For example, the open and closing curly braces `{ }` represent a namespace. See the topic "Class View and Object Browser Icons" in the Visual Studio documentation for a complete list of the icons used.

5. **Click one of the public methods.**

   The method's source code appears in the Code Definition window, as shown in Figure 2-9.



```
public struct RAW_READ_INFO
{
    public long DiskOffset;
    public uint SectorCount;
    public uint TrackMode;
}
public struct CD_TRACK
{
    public uint Address;
    public uint Length;
}
public class CDAudio
{
    public const uint RAW_SECTOR_SIZE = 2352;
    public const uint CD_SECTOR_SIZE = 2048;
    public const uint MAXIMUM_NUMBER_TRACKS = 100;
    public const uint SECTORS_AT_READ = 20;
    public const uint CD_BLOCKS_PER_SECOND = 75;
    public const uint IOCTL_CDROM_RAW_READ = 0x2403E;
    public const uint IOCTL_CDROM_READ_TOC = 0x24000;

    //#define IOCTL_STORAGE_MEDIA_REMOVAL          CTL_CODE(IOCTL_STORAGE_BASE, 0x0
```

Figure 2-9: Source code appears in the Code Definition window.

The Code Definition window is a read-only view of the source code. To open the source code file in the Code Editor, right-click the item in the Class View window and choose Go to Definition from the contextual menu.

To view the object in Object Browser, right-click the object in the Class View window and choose Browse Definition from the contextual menu.

Accessing Types in Your Source Code

Data types are logically organized into namespaces but physically organized into assemblies. In order to consume types in your source code, you need to know how to access the physical and logical paths to data types.

- **References** provide access to the physical assembly files where the types can be found.
- **Namespaces** provide access to the logical path of the type within the referenced assembly.

You access the physical file where types are stored by adding a reference to the type. Visual Studio provides the ability to add new references and manage existing references in your project.

When you create a new project, Visual Studio automatically adds a number of references to common, physical DLL files you may need to use in your project. You might remove any reference that you aren't using.

After you add a reference to the physical assembly where the type resides, you must also supply the logical namespace to access the type and its members. You can type the namespace in source code. If you have properly referenced the assembly, IntelliSense pops up to help you find the namespaces you need.

You can also include namespace directives — at the top of your source code — to provide a shortcut to namespaces you wish to use in your source code. The namespace directive in C# is `using`, and it's `Include` in Visual Basic. These directives allow you to access types within a namespace without fully qualifying the namespace every time.

For example, say you want to create a new `DataSet` in your source code. The `DataSet` type is found in the `System.Data` namespace. Using the fully qualified namespace looks like this in C#:

```
System.Data.DataSet ds = new System.Data.DataSet();
```

However, if you add the following using directive at the top of your source code file

```
using System.Data;
```

you can create a new DataSet:

```
DataSet ds = new DataSet();
```

Using namespace directives often saves space in your code and makes your code easier to read.

Another approach is to create an alias for the namespace so that it quickly identifies that the type is being referenced from another namespace. To create an alias in C#, type the following:

```
using data = System.Data;
```

Now, you can qualify the type by using the alias, as shown in the following code:

```
data.DataSet ds = new data.DataSet();
```

The same code in Visual Basic appears as

```
Imports data = System.Data
Dim ds as new data.DataSet
```

Chapter 3: Get Some Class

In This Chapter

- **Using objects and classes**
- **Declaring classes and members**
- **Designing classes with Class Designer**

When I first learned how to program computers, the flow of programming was sequential. You wrote one line of code after another. Each line of code was numbered, and you thought about your program in terms of what happens first and what happens next.

When I went to college, I was taught a more sophisticated approach to programming that uses functions to group statements of code. Functions are great because you can easily organize your code into reusable blocks.

When I started working as a programmer, I encountered yet a third programming paradigm that organizes code into classes. I found this curious because even though a program's basic building block is classes, the style of programming is called object-oriented programming (OOP).

Visual Studio provides many tools that support the activities of object-oriented design (OOD) and programming. This chapter covers some of the basics of working with objects and classes and shows you how to use Class Designer to visually design and inspect classes and objects.

Bringing Programs to Life the Object-Oriented Way

Most programmers take awhile to figure out the relationship between objects and classes when they first encounter OOP. If you think of software as a means of modeling a real-world scenario, the following points are worth remembering:

- ***Objects* represent the people, places, and things in your program.** For example, think about a business that sends an invoice to a customer. Invoice #100984 and Customer #60093 are objects that your program needs to manipulate. The business, ABC Graphics, sending invoice #100984 to customer #60093 is an object, too.

- ***Classes* are the units of code you write to make objects come to life in your program.** For example, your program defines `Invoice` and `Customer` classes. Each class encapsulates the data and behaviors required to create objects that represent real-world invoices and customers.

You *write* classes, but you *think* in terms of objects, which is why programming with objects is called *object-oriented programming.* Table 3-1 compares OOP with procedural and sequential programming styles.

Table 3-1 Comparison of Programming Styles

Programming Style	*Programmers' Approach*	*Basic Building Blocks*
Object-oriented	What objects am I modeling in my program?	Objects
Procedural	What functions do I need to tackle this problem?	Functions
Sequential	What's the next line of code to write?	Statements

Thinking in objects isn't limited to programmers. Objects are often found in the planning and design stages of a project, where object-oriented thinking is formally known as both of the following:

- **Object-oriented analysis:** Requirements analysts use objects to model problems in the business domain.
- **Object-oriented design (OOD):** Designers use analysis objects to model classes that programmers will write to create objects.

Programmers use object-oriented programming languages (OOPLs) — such as Visual Basic, C#, C++, and Java — to implement classes.

Even if you've never written an object-oriented program or know nothing about objects, chances are you're familiar with several of the characteristics of OOP:

- Using the familiar dot notation to access the members of an object
- Using the new operator to initialize a reference type
- Setting an object's properties

What's with Objects?

Classes and objects are the bedrock of OOP. *Classes* are special kinds of data types that you use to create objects in your code. Objects are placeholders in your code for the real people, places, and things upon which your program acts.

Modeling data with classes

Classes are a complex data type. Simple data types store values, such as integers and Booleans. Composite data types, such as structures that are capable of storing more than one value, aren't more sophisticated than integers and Booleans. The values that you can store by using classes, however, are beyond mere integers or sets of integers.

You might recognize simple data types as value types and complex data types as reference types. See Chapter 2 in Book III for a thorough discussion of the differences between value types and reference types.

A *class* is a data type that's capable of storing data and actions. When you declare and initialize a variable by using a class data type, you create an object that stores data and action values defined by the class data type. Objects are used to accommodate the potential complexity of values stored for class data types. An object is often referred to as an *instance* of the class data type.

Book III
Chapter 3

Get Some Class

The term *object* is often interchanged for the term *class.* People often use the term *object* in a generic sense to mean "the class that defines the object." People seldom use *class* when they mean *object.* You must consider the context in which the term is used to determine whether the author or speaker means class or object. (Ask whether the author is writing about a data type or the value represented by the data type. If it's the former, it's a *class.* Otherwise, it's an *object.)*

Classes act as a template for defining the data and actions of an object. You create objects by using the class as a template. Each object created from the class is like a blank form into which you can place data. The source of that data may be a database, data typed on a screen by a user, or another object.

Some people find it helpful to think of classes as the blueprints for objects — or to think that classes are abstract and that objects are concrete.

Think of classes as a template for creating an object.

Create objects to store data

You're familiar with the concept of data, but you may not immediately understand what the potential actions of a class data type may be. Consider a class data type called `Customer`. You expect a `Customer` data type to store a customer's name and account number. If you need to tell someone a customer's 30-days aging of their account, how might you approach solving this problem?

Given that an account's aging is based on the current date, you need to be able to calculate the value in real time. You can't query a database for the answer. Your `Customer` class needs a procedure that returns the account's aging.

TIP

Storing calculated values in a database is usually considered bad form. There are a number of reasons that you don't want to store calculated values, but the two most important reasons are that the calculated values consume additional space and they can become outdated when the values that support them change. Classes often return calculated values for an object. For example, an `Employee` class may have a procedure that returns the employee's age.

Another way to think of actions is that actions are sometimes things that an object can do. For example, a student registers for courses. Registering for courses is an action that a student object is capable of doing.

Classes are important to software development because they

- Are a basic building block of programs
- Group related code in a unit
- Control visibility and accessibility of your code
- Encourage reuse
- Are easy to extend
- Can model real-world scenarios
- Map code to the real world

Code created with objects and classes, adhering to the principles of OOP, display these characteristics:

- **Abstraction:** Models a real-life thing by using only the attributes relevant to solving the problem at hand.

Creating user-defined classes is an example of an abstraction. When you approach a problem you're trying to solve, you identify the objects that are relevant to the problem. For example, you need a `Customer` object and an `Invoice` object to prepare invoices. You determine the level of detail required in your `Customer` and `Invoice` classes to adequately solve the problem of printing invoices.

- **Encapsulation:** Wraps the data and actions of an entity into a single unit.

  Encapsulation isn't unique to OOP. Functions and subroutines are examples of encapsulation, as are classes.

- **Inheritance:** Builds a hierarchy of related code by creating parent-child relationships between units of code.

  Inheritance allows you to create a supertype/subtype relationship between types. A simple example of inheritance is having a `Dog` class and a `Cat` class that inherits from an `Animal` class. The `Animal` class has all the data and actions common to both dogs and cats. However, unique behaviors — such as a cat's purr — appear in the appropriate derived class.

- **Polymorphism:** Allows child types to provide their own code to implement actions performed by their parent or sibling.

  Polymorphism makes it possible for a class hierarchy to share a common set of behaviors while each has its own implementation. For example, the `Animal` class has a reproductive cycle. The `Dog` and `Cat` classes each require their own implementation of a reproductive cycle. Polymorphism allows you to refer to the generic concept of a reproductive cycle in your code, regardless of whether you mean a dog or a cat.

Another important characteristic of OOD is information hiding. You should design your classes so that important design decisions are hidden. In other words, one should have to think about only initializing and accessing your classes, not understanding how you've written them. The classes in the .NET Framework are a good example of information hiding. You must explicitly think about information hiding as you design your application. It's possible to abstract, encapsulate, inherit, and polymorph without hiding information. Simply using object-oriented techniques doesn't ensure that you're hiding information.

You use an OOPL (such as Visual Basic, C#, or C++) to write object-oriented programs. The .NET Framework is an object-oriented library of classes.

In addition to the OOP languages, Visual Studio provides several tools to support OOP, including

- Class Designer
- Class View
- Object Browser
- Object Test Bench (OTB)

The only way to learn OOD and programming is to do OOD and programming. Memorizing terms and definitions can help your vocabulary, but by themselves, they do very little to increase your understanding of OOD and OOP. So, instead of getting hung up on terminology, just start coding.

Anatomy of a Class in .NET

Classes and OOP have terminology with which you might not be familiar and which might seem overwhelming at first. At its heart, classes are just ways to organize code similar to the modules, functions, and subroutines with which you might be familiar. Like modules, functions, and subroutines, you declare variables and procedures similarly to how you do in modules.

Classes are templates that define an object's

- **Data:** Data is sometimes referred to as the object's *state,* which is the set of values an object holds at a given point in time.
- **Behavior:** Behaviors are the actions that the object takes.

The code that creates a new class is a *class declaration.* A class declaration consists of the class header and body. The class header defines the following:

- **Attributes:** Optional keywords used by the compiler. The .NET Framework provides several attributes for use in your class and method declarations.

  Chapter 6 of Book III shows you how to use the `WebService` and `WebMethod` attributes to turn ordinary classes and methods into Web services.
- **Modifiers:** Keywords that define how the class may be used. For example, classes and members use access modifiers, such as `public` and `private`, to determine whether the class or member may be accessed by code outside the class or namespace.
- **Name:** Identity of the class.
- **Base class:** Data type from which the class is inheriting. All classes implicitly inherit from `System.Object`.

- **Interfaces:** Comma-separated list of interfaces implemented by the class. An *interface* is a set of member declarations with no source code to implement them. Interfaces make it possible to define a publicly consumable contract without writing the source code. Other developers can target the interface without fear of the interface changing. See Chapter 4 of Book III for more information on interfaces.

A class' *body* defines the class' data and behavior. The items declared inside a class are its *members.* The members you can create inside a class body include

- **Constants:** Values in the class that don't change.
- **Constructors:** Special procedures that are called to create an object using the class. An object is often referred to as an *instance* of the class.
- **Destructors:** Special procedures that are called before an object is discarded by your program.
- **Events:** Procedures that raise a notification when a certain action occurs.
- **Fields:** Variables declared for use within the class, often for storing data.
- **Indexers:** Procedures that index individual instances of the class.
- **Methods:** Procedures that provide the behaviors of the class.
- **Operators:** Procedures that define conversions for the class and extend built-in operators.
- **Properties:** Procedures that provide access to the class' data structure.
- **Types:** Nested data types, such as classes or structures, created within the class.

The combination of a class header and its body is its *definition.*

You may have noticed that a class' body is made up of procedures and variables. If you've done any programming at all, you're probably familiar with the concept of using variables and procedures to create your program.

In C#, a class' body is wrapped in opening and closing curly braces. Listed here is a class definition in C#:

```
public class Message
{
  private string m_message;
  public string Contents
  {
    get { return m_message; }
    set { m_message = value; }
  }
```

```
    public Message(string message)
    {
      this.m_message = message;
    }
    public string ReverseContents()
    {
      char[] c = this.m_message.ToCharArray();
      StringBuilder sb = new StringBuilder();
      for (int i = 1; i <= c.Length; i++)
      {
        sb.Append(c, c.Length - i, 1);
      }
      return sb.ToString();
    }
  }
```

Can you tell which elements in this class are fields, properties, and methods? It's not immediately obvious just by looking at the code. To someone not familiar with OOP, this code looks like variable and procedure declarations. Table 3-2 summarizes the members in this class.

Table 3-2 Members Found in the Message Class

Member Type	*Quantity*	*Example*
Field	1	`m_message`
Property	1	`Contents`
Constructor	1	`Message(string message)`
Method	1	`ReverseContents()`

Take a look at the equivalent code sample in Visual Basic. Notice how the Visual Basic keywords `Property` and `New` provide clues as to the member's purpose:

```
Public Class Message
  Private m_message As String
  Public Property Contents() As String
    Get
      Return Me.m_message
    End Get
    Set(ByVal value As String)
      Me.m_message = value
    End Set
  End Property
  Public Sub New(ByVal message As String)
    Me.m_message = message
  End Sub
  Public Function ReverseContents() As String
    Dim c As Char() = Me.m_message.ToCharArray()
    Dim sb As New StringBuilder
    For i As Integer = 1 To c.Length
      sb.Append(c, c.Length - i, 1)
    Next
    Return sb.ToString()
  End Function
End Class
```

Classes are logically organized within namespaces. Your class code is stored in a physical file called an *assembly.* A single namespace can span multiple assembly files, but a single class can reside in only one assembly. To access classes, you must reference the assembly and the namespace within your project. See Chapter 2 of Book III for more information.

TIP

The C# keyword `this` and Visual Basic keyword `me` are used to reference the current instance of an object in your code.

Inheriting the services of System.Object

All classes inherit from the .NET class `System.Object`. Inheriting from `System.Object` provides all the basic services your classes need to function as objects and get along with other objects.

`System.Object` is the base class of your class, and your class is a derived class of `System.Object`.

Any time your classes derive from a base class, you should check the documentation of the base class to determine whether there are any methods you should override. When you *override* a member, you write your own code to implement the member.

For example, the `System.Object` class has a method called `ToString` that you should override. By overriding `ToString` in the classes you define, you can get a string representation of objects created from your class. See Chapter 5 of Book III for more information on overriding.

Several overridden methods are available for `System.Object`. Table 3-3 lists all the methods your class inherits from `System.Object` and identifies those that you should override.

TIP

Overriding a base class' methods when recommended is considered good form. Overriding ensures that your derived class works as expected. Look for the "Note to Implementers" section in the base class' type for information on overriding.

Table 3-3 Methods of System.Object

Method	*Description*	*Override*
`New`	Constructor	Not required
`Finalize`	Destructor	Not recommended
`Equals`	Determines whether two object instances are equivalent	Yes
`GetHashCode`	Generates a hash code for use in hashtables	Not required
`GetType`	Returns the data type of the object instance	No
`MemberwiseClone`	Copies a value or an object reference	No
`ReferenceEquals`	Determines whether two object instances are the same object	Not required
`ToString`	Returns a human-readable representation of the object instance	Recommended

Using classes in your code

Because classes are data types, you use them like you use any data type. They're reference types, so you use the `new` operator to create a new instance of a class.

The `new` operator calls the class's constructor method. A class may have more than one constructor. If no constructor method is defined in the class, the object is initialized by using the default constructor. The default constructor is the `New()` method defined in the `System.Object` class. Recall that all classes derive from `System.Object`.

Classes are reference types, so everything about how memory is allocated for reference types in Chapter 2 of Book III applies to classes. When you initialize a new instance of a class, the object itself is allocated on the heap. The reference to the object is allocated on the stack.

To create a new instance of the `Message` class created in the preceding section, you type the following:

```
Message m = new Message("Hello");
```

You use dot notation to access the members of the class. The member's visibility is determined by the modifier set on the member's declaration. You can always see public members, but not private members. To call the `ReverseContents` member, type the following:

```
string s = m.ReverseContents();
```

Here's the equivalent code, using Visual Basic syntax:

```
Dim m As New Message("Hello")
Dim s As String = m.ReverseContents()
```

TIP

You don't need to declare the `Message` class using Visual Basic and C# syntax. You can just as easily create the class in C# and then reference and instantiate it in Visual Basic.

Hopefully, this syntax looks familiar to you. If you've used any features in the .NET Framework, you've encountered this syntax.

TIP

You can use any of the classes in the .NET Framework or any third-party library just by knowing the rules for declaring variables based on a class data type and creating instances of objects.

Using Class Designer

Visual Studio provides a visual design tool — Class Designer — that you can use to create class diagrams. Some common uses of Class Designer are

- ✦ Designing new classes
- ✦ Visualizing existing classes
- ✦ Refactoring classes

Another use of Class Designer that might not be immediately evident is as a training aid. You can use Class Designer to familiarize yourself with the concepts of OOP. You can also use it to learn syntax.

WARNING!

Use Class Designer as an aid, and not a crutch. If you rely too heavily on the tool, you risk being dependent on it. Becoming dependent on the Class Designer is definitely not where you want to be if your company decides to migrate some projects to another object-oriented language, such as Java.

Like most of the visual designers in Visual Studio, Class Designer generates diagrams from code and code from diagrams. The diagrams and code are synchronized. The class diagram is a visual representation of source code.

TIP

The class diagrams created by Class Designer are design diagrams — not analysis diagrams. Design diagrams usually show more implementation details than analysis diagrams. If you want a tool better suited to creating analysis diagrams, try Visio for Enterprise Architects. You can easily generate classes from your Visio diagrams and then view those classes in Class Designer. You can find out more about Visio by reading *Visio 2007 For Dummies* (Wiley) by John Mueller and Debbie Walkowski.

Common tasks you can do in Class Designer include the following:

- Create new classes and members.
- Define relationships among classes.
- View classes and relationships.
- Refactor code.

Exploring Class Designer

Class Designer is capable of designing more than just classes. You can use Class Designer to design or view any kind of data type, including

- Classes
- Delegates
- Enumerations
- Interfaces
- Structures

Class Designer can also create Visual Basic modules, which you can use to encapsulate your code. See the Visual Basic documentation for more information on the differences between classes and modules.

You add a new class diagram to your project using the Add New Item window. A project may have multiple class diagrams. To add a new class diagram

1. **Right-click your project's folder in Solution Explorer.**
2. **From the contextual menu that appears, choose Add⇨New Item.**

   The Add New Item window appears.

3. **Click the Class Diagram icon.**
4. **Type a name for the class diagram in the Name field.**

   The example uses `ClassDiagram1` as the class diagram name.
5. **Click the Add button.**

   Visual Studio adds the class diagram file to Solution Explorer and opens it in Class Designer.

Class diagrams use the `.cd` filename extensions. Like most files in Visual Studio, the class diagram file is an XML file.

Class diagrams are visual representations of classes in a project. The diagrams are similar to diagrams created using the Unified Modeling Language (UML) specification.

UML is a modeling language that identifies 13 diagrams used for designing software. Many tools available on the market are capable of creating UML diagrams, including Microsoft Visio for Enterprise Architects. Visual Studio class diagrams won't replace UML diagrams you're already using. Consider them another tool to add to your modeling arsenal.

Class Designer consists of the

- Toolbox
- Designer surface
- Class Details pane

The Designer provides a toolbar and shortcut menus to access additional features. The Designer works with Solution Explorer, Class View, and OTB.

Class Designer provides several commands you can use for working with your diagram. The commands that you can use depend on what you have selected in the diagram. You have three levels of commands for working with class diagrams:

- **Diagram:** To work with the entire diagram, click a blank area of the diagram.
- **Type:** Click a shape to access commands that act on the data type.
- **Member:** Click a member within the type to view commands.

Figure 3-1 shows the Class Designer toolbar. The first column of Table 3-4 lists the commands of the Class Designer toolbar, as they appear on the toolbar from left to right.

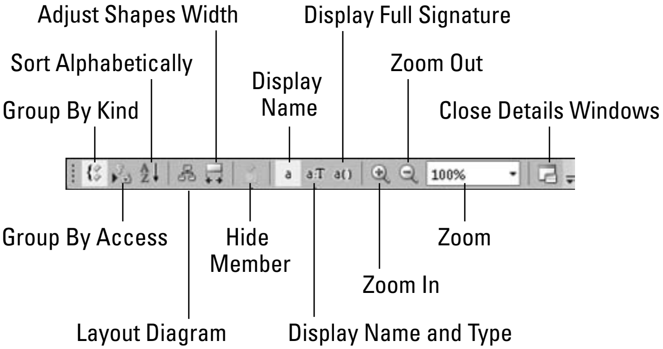


Figure 3-1: Use the Class Designer toolbar to control the diagram's layout.

Table 3-4 Class Designer Toolbar Commands

Command Type	*Command*	*Description*
Arrangements of shapes	Group by Kind	Groups by kind of data type
	Group by Access	Groups by data type's access modifier
	Sort Alphabetically	Sorts by data type's names
Layout	Layout Diagram	Organizes diagram
	Adjust Shapes Width	Widens shapes
Member display options	Display Name	Displays members' names only
	Display Name and Type	Displays names and type
	Display Full Signature	Displays full declaration
Visual	Zoom In	Zooms in on diagram
	Zoom Out	Zooms out on diagram
	Zoom Percentage	Sets a zoom percentage
	Class Details Window	Opens the Class Details window

The Class Designer menu provides you the opportunity to act upon the shapes that represent data types on the diagram. Table 3-5 lists the options available for data types you click in the class diagram.

Table 3-5 Class Designer Menu Type-Level Commands

Command	*Description*
Add	Selects a member to add to the type
Refactor	Accesses refactoring options for the type
IntelliSense	Generates code using IntelliSense
Show Base Class	Displays the type's base type on the diagram
Show Derived Class	Displays types derived from the type on the diagram
Collapse	Collapses members
Expand	Expands members
Show All Members	Displays any hidden members
Create Instance	Creates an object using the OTB
Invoke Static Method	Calls a static method using the OTB

You can access most of these commands from a shortcut menu when you right-click a shape in a class diagram. The shape's shortcut menu includes commands to delete the shape from the diagram and view the type's underlying source code.

Designing classes

One of the more exciting features of Class Designer is the ability to design classes. You can drag and drop data types on the class diagram; in the background, Visual Studio is generating the code to create the data type.

Creating a new class

Class Designer can create all kinds of data types. The Toolbox includes all the shapes you need to create data types and show relationships between data types.

To create a new class with Class Designer, follow these steps:

1. **Add a new class diagram, following the steps in the previous section.**
2. **Press Ctrl+Alt+X to open the Toolbox.**

   The Toolbox pane displays icons representing items you can add to the Class Designer. Open the Class Designer tab, if necessary, to see the controls you can use.

3. **Drag and drop a Class icon from the Toolbox onto Class Designer.**

   The New Class dialog box appears.

4. **Type a name for the class in the Name text box.**

   The example uses Message as the name.

5. **Select an access modifier from the Access drop-down list.**

   The `public` access modifier is accepted by default. The example uses `public` as the access modifier.

6. **Type a new filename for the class or select an existing file to add the class.**

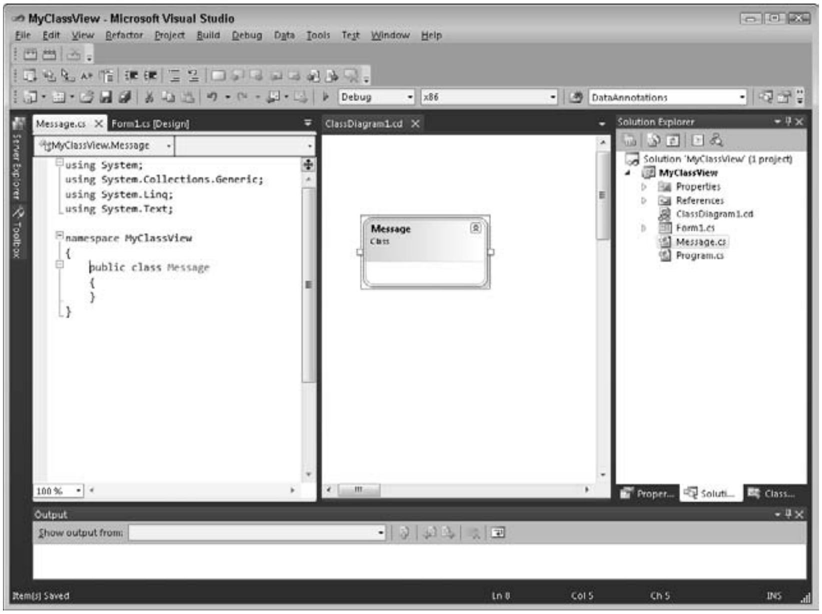
   The example uses the Create New File option and `Message.cs` as the filename.

7. **Click OK.**

   The Class shape appears on Class Designer.

   Visual Studio generates the class declaration and places it in the file you created or selected in Step 6.

Figure 3-2 shows Class Designer with a class called `Message`. You can see the `Message.cs` source code that Visual Studio created in the pane next to the class diagram.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MyClassView
{
    public class Message
    {
    }
}
```

Figure 3-2: Visual Studio generates the class declaration for the new class.

To add more types to the class diagram, drag the type from the Toolbox onto the design surface. A dialog box, similar to the one in Step 3, appears.

Adding members to a class

You use Class Designer to add members to your class. When you add a member using the designer, you specify the following information about a member:

- Name
- Data type
- Access modifier

You can, optionally, supply a descriptive comment and mark whether to hide the member on the class diagram.

Visual Studio uses the information you supply to create a member declaration in your source code. You have two options for adding members to your class with Class Designer:

- **Type the members in the Class Details pane.** The Class Details pane shows the class's members grouped by methods, properties, fields, and events.
- **Add the members to the class in the class diagram.**

The class diagram and the source code are synchronized. Any members you declare directly in source code appear in the class diagram.

To add a new member via the Class Details pane

1. **Click the section of the Class Details pane for the type of member you wish to add.**

   If the Class Details pane is closed, choose View➪Other Windows➪Class Details to open the pane.

2. **Type the member's name in the line immediately below the section's header.**

   For example, to add a new field, click the line below the Fields section where you see `<add field>`.

3. **Press the Tab key.**

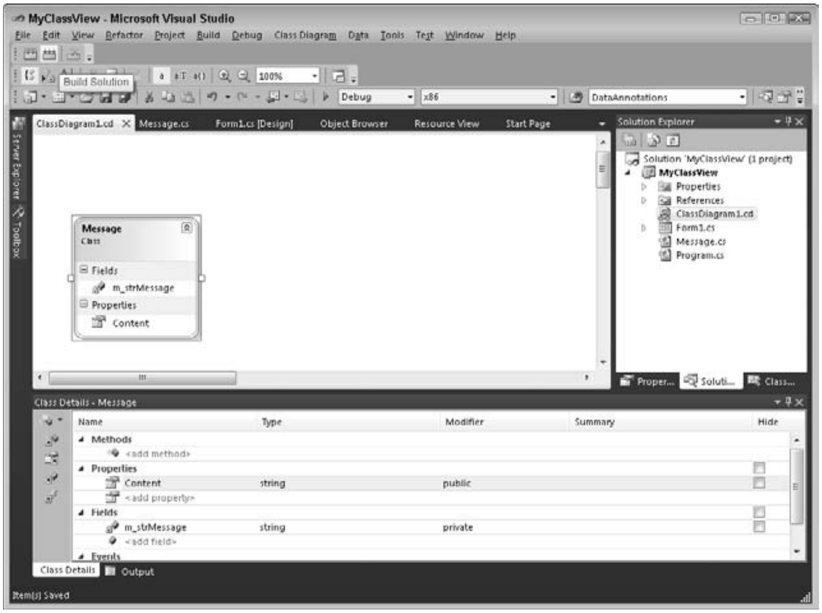
   The cursor stops in the Type cell.

4. **Type the member's data type.**

   IntelliSense works in the Class Details pane.

5. **Press Tab and select the member's access modifier.**
6. **Press Tab and add an optional descriptive comment.**
7. **Click to place a check mark in the Hide column if you don't want the member to appear on the class diagram.**

   Figure 3-3 shows a field (`m_strMessage`) and a property (`Content`) in the Class Details pane.



Name	Type	Modifier	Summary	Hide
Methods				
<add method>				
Properties				□
Content	string	public		□
<add property>				
Fields				□
m_strMessage	string	private		□
<add field>				
Events				

Figure 3-3: Use the Class Details pane to add new members.

Visual Studio generates the member declarations for you in the class's source code file.

The Class Details pane features a toolbar button with shortcuts for adding members. To access the shortcut button, click the arrow on the button, as shown in Figure 3-4.

Adding a new constructor method

A method's declaration is a little more complex than a field or property declaration because a method may accept parameters. In addition, multiple kinds of methods are available, such as constructors and destructors. The Class Details pane has no problems adding methods to your class.

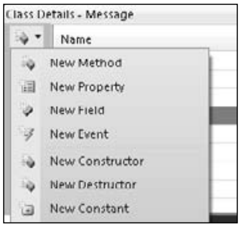


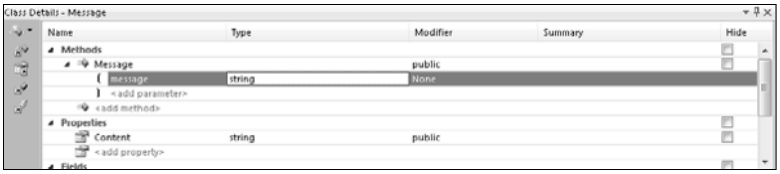
Figure 3-4: Use the Class Details toolbar to add members.

To add a new constructor method, follow these simple steps:

1. **Click the shortcut button on the Class Details toolbar.**
2. **Chose New Constructor.**

   A new constructor method appears in the Methods section of the Class Details pane.

   Constructors use the same name as the class in C#. In Visual Basic, constructors are named `New`.
3. **Type the constructor's parameters in the lines below the constructor's declaration, as shown in Figure 3-5.**


Class Details - Message

Name	Type	Modifier	Summary	Hide
◢ Methods				□
◢ Message		public		□
(message	string	None		
) <add parameter>				
<add method>				
◢ Properties				□
Content	string	public		□
<add property>				
◢ Fields				□

Figure 3-5: Add parameters to the constructor method.

Visual Studio generates the member declarations and stubs out the member's bodies. (The members don't perform any useful task — they simply provide a placeholder for code you add later.) Exceptions are placed inside the generated members to remind you that you need to add your code if you try to use the members. For example, the following code is the constructor generated for the preceding sample:

```
public Message(string message)
{
  throw new System.NotImplementedException();
}
```

TIP

Class Designer is a good tool to use for designing a class domain. Designers can lay out the classes and then pass off their implementation to a programmer.

Creating members using the class diagram

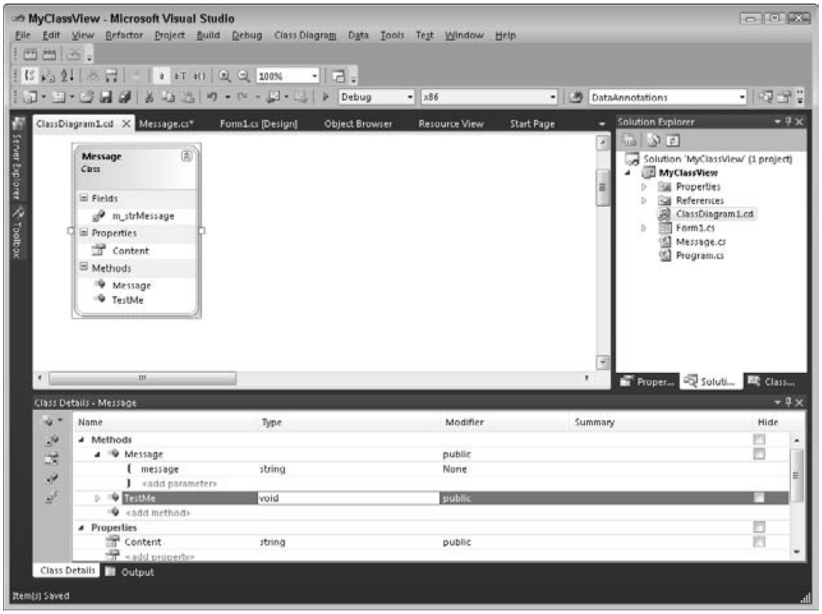
You can also create members by using the class diagram. To create a member in the class diagram, do the following:

1. **Right-click the class in the class diagram.**
2. **From the contextual menu that appears, choose Add.**

   A list of members appears, which you can add to the class.
3. **Select the type of member you wish to add.**

   The member appears in the diagram.
4. **Type the member's name in the diagram, as shown in Figure 3-6.**

   Use the Class Details pane to complete the member's declaration.



Name	Type	Modifier	Summary	Hide
Methods				□
Message		public		□
(message	string	None		
) <add parameter>				
TestMe	void	public		□
<add method>				
Properties				□
Content	string	public		□
<add property>				

Figure 3-6: Type the member's name in the class diagram.

Creating associations

The nature of OOP is that classes interact. You create an interaction between classes when you create member declarations in your class. For example, if you declare a field of the string type, you create an association between your class and the string class.

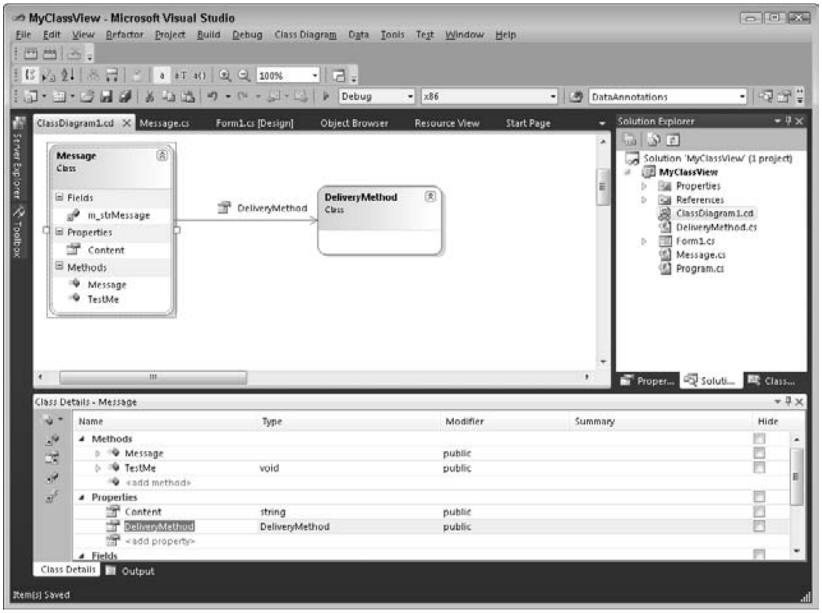
In Class Designer, you show associations between only classes that are relevant to the model you're creating. You could show every association, but then your diagrams would be too crowded for people to understand. Instead, you should show only the associations that you want to draw attention to.

Here's how to create an association between two types:

1. **Create both types on the class diagram.**
2. **Click the Association icon in the Toolbox.**
3. **Click the type where you want to draw the association.**

   The cursor appears as a line, as shown in Figure 3-7.
4. **Drag the line to the class.**

   An association between the classes appears. Visual Studio adds the association as a property.



Name	Type	Modifier	Summary	Hide
Methods				□
Message		public		□
TestMe	void	public		□
<add method>				
Properties				□
Content	string	public		□
DeliveryMethod	DeliveryMethod	public		□
<add property>				
Fields				□

Figure 3-7: Drag the cursor between the two classes to create an association.

You can show associations between classes and collections. To show an association between a class and a strongly typed collection, follow these steps:

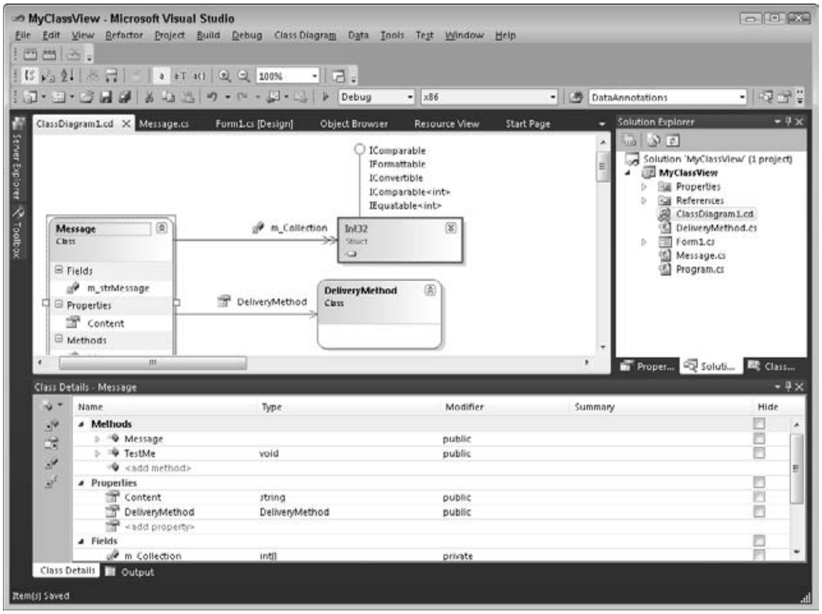
1. **Add the member to your class that uses the strongly typed collection.**

   For example, add a field that uses a generic collection based on another data type in your diagram. The example uses a field named `m_Collection`, which is of type `Int32[]` (an array).

2. **Right-click the member on the class diagram.**

3. **From the contextual menu that appears, choose Show as Collection Association.**

   An association between the classes appears, as shown in Figure 3-8.



Name	Type	Modifier	Summary	Hide
Methods				□
Message		public		□
TestMe	void	public		□
<add method>				
Properties				□
Content	string	public		□
DeliveryMethod	DeliveryMethod	public		□
<add property>				
Fields				□
m_Collection	int[]	private		□

Figure 3-8: Create an association between a class and a collection.

Generating code

Class Designer takes advantage of the Visual Studio code generation features. While using Class Designer, you can access

- **Advanced IntelliSense features,** such as overriding type members and implementing abstract base classes.
- **Refactoring commands,** such as extracting interfaces from a type and renaming members.

Refactoring commands aren't available while designing classes using Visual Basic.

See Chapter 5 in Book III for more information on using IntelliSense and refactoring features of Visual Studio.

Viewing existing classes

Class Designer is an excellent tool for viewing information about data types. You can view data types within your own project, as well as any types referenced by your project. This means that you can view types from the .NET Framework or a third party.

To view a type in your project, simply drag and drop the source code file from Solution Explorer to the class diagram.

You use Class View to drag referenced types from your project to the class diagram. For example, to add the type `NullReferenceException` to a class diagram, follow these steps:

1. **Add a new class diagram to your project.**
2. **Press Ctrl+Shift+C to open Class View.**
3. **Type** System.NullReferenceException **in the Search box.**

   Alternatively, expand the Project Reference folders and navigate to the `NullReferenceException` type.

   The `NullReferenceException` type is in the `System` namespace in the `mscorlib` assembly. Your project references the `mscorlib` assembly by default.
4. **Press Enter.**

   The `System.NullReferenceException` appears in the search results.
5. **Drag and drop System.NullReferenceException on the class diagram.**

Admittedly, viewing a single type on a class diagram isn't very exciting. You can use the Class Details pane to view the type's members. You can also view a type's inheritance hierarchy.

To view a type's base type, follow these steps:

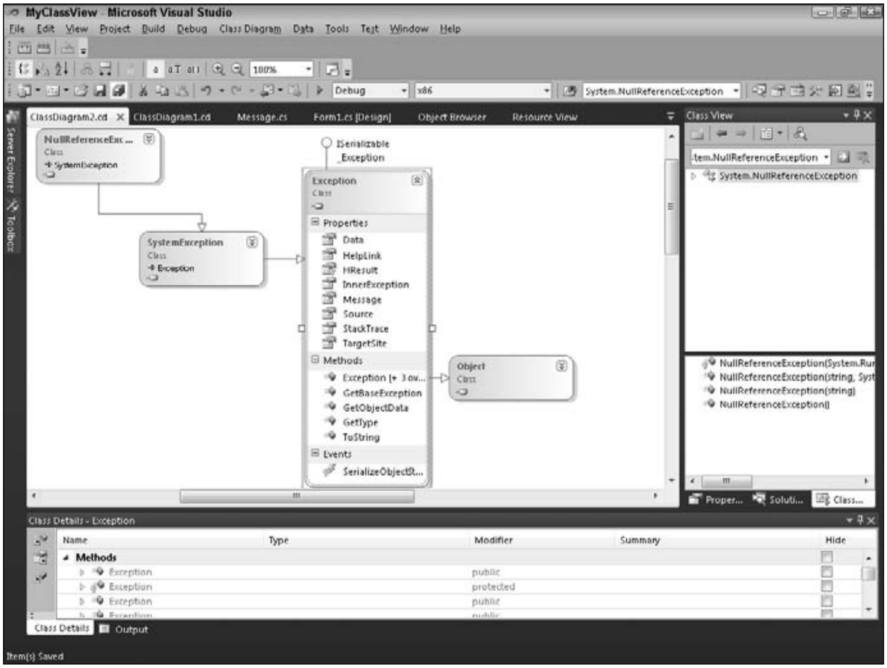
1. **Right-click the type and choose Show Base Class from the shortcut menu.**

   The base type appears on the class diagram.

2. **Repeat Step 1 for each base type you display.**

   Show Base Class appears dimmed when you reach the top of the inheritance hierarchy.

   Figure 3-9 shows the class diagram for the `NullReferenceException` type. Notice that `System.Object` is at the top of the hierarchy.



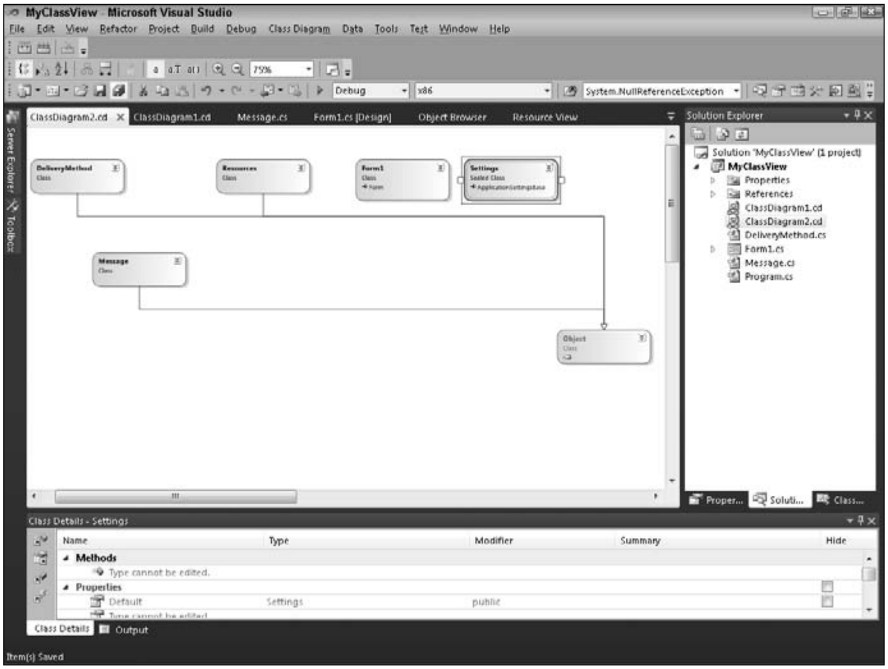
Name	Type	Modifier	Summary	Hide
Methods				□
Exception		public		□
Exception		protected		□
Exception		public		□
Exception		public		□

Figure 3-9: Use Class Designer to view a type's inheritance hierarchy.

To view the class diagram for an entire project, follow these steps:

1. **Right-click the project folder in Solution Explorer.**
2. **From the contextual menu that appears, choose View Class Diagram.**

   A class diagram of the project appears. Figure 3-10 shows the class diagram for this example (which started out as a Windows Form application).



Name	Type	Modifier	Summary	Hide
Methods				
Type cannot be edited.				
Properties				□
Default	Settings	public		□
Type cannot be edited.				

Figure 3-10: Create a class diagram for a project.

Chapter 4: Wiring Up Events in the User Interface

In This Chapter

- **Understanding events and event handlers**
- **Using Visual Studio to discover events**
- **Creating event handlers with Visual Studio**
- **Getting a grip on Windows and Web applications life cycles**

Code fires in response to events, such as button clicks. Taking advantage of the events in an application and control life cycles allows you to run code at a precise moment in an application's or a control's life cycle. Instead of waiting for a user to click a button, you can run code as soon as your application starts — or even when a user's cursor leaves a text box.

In Visual Studio, responding to events is as simple as a double-click, in many cases. Visual Studio generates codes behind the scenes that wire up your code to the event that fires it.

Handling Events

In most modern applications, an application responds to actions that a user takes. Your application sits in a waiting state until a user comes along and clicks a button, a menu, or a drop-down list. When a user clicks a button, an action (the *event*) occurs. The code that executes when an event fires is an *event handler.*

Procedures that run when an event occurs — *event handlers* — um, handle events. Still, they're just procedures, just like any procedure you write. You're responsible for writing event handlers to handle any events to which you wish to respond.

For example, say that you want a message box to pop up when a user clicks OK on your Windows Form. The following code sample uses Visual Basic syntax to make a message box appear:

```
MessageBox.Show("You clicked the OK button", "Handling Events",
    MessageBoxButtons.OK)
```

Now, the question is where to put that code. You know that you want the message box to appear when the user clicks a button. You need to write an event handler for the button's `Click` event.

Right here is where you should stop. How do you know that the code belongs in the button's `Click` event? Is it as simple as pulling the verb out of a sentence and calling it an *event?* Not quite.

Discovering events

Most documentation and articles that you can read about events and event handling assume that readers somehow magically know which events they're supposed to write code for. Most people intuitively get the concepts of *click* or *double-click,* but what about *hover* or *paint?*

In order for a button or some other component to have an event, someone must have written an event method for the component. The event method is the other side of the event-handling story.

For example, a button that you add to a Windows Form is actually the `Button` class in the `System.Windows.Forms` namespace. The button's `Click` event is the `Button` class' `Click` event method. An *event method* is a procedure in the same way that an event handler is a procedure. The difference is that the event method creates the event, whereas the event handler responds to the event.

The class with the event method is the *sender,* and the class with the event handler is the *receiver.* When an event fires, the sender *raises* the event. The receiver *consumes* the event with an event handler.

By reducing events to methods in a class, all you have to do is refer to the class' documentation to figure out the set of events available to you. Of course, Visual Studio has a few tools you can use to help you in your quest:

- **The Properties window:** View the Properties window of a component and click the lightning bolt icon to see a list of the component's events.
- **The Code Editor Navigation bar:** Use the Navigation bar at the top of the Code Editor to select an event.
- **Help documentation:** Use the index or the Class Library Reference in the .NET Framework software development kit (SDK) to locate documentation on a specific class.

Adding a component

The easiest way to discover events is to add a component to Windows Forms Designer or Web Forms Designer and view its properties. Here's how:

1. **Create a new Windows or Web project or open an existing project.**

2. **Drag and drop a control or component from the toolbox onto the form designer.**

   The control or component appears in the designer.

   Press Ctrl+Alt+X to open the toolbox if it's closed.

3. **Click the control or component to select it.**

4. **Press F4 to open the Properties window.**

   The control or component's properties appear in the Properties window.

5. **Click the Events button on the Properties window toolbar to display a list of events available for the control or component.**

   The Events button is the button with the lightning bolt.

   The lightning bolt is the icon used to represent events in Visual Studio. You'll see the lightning bolt used in Object Browser, Class View, and the Code Editor Navigation bar.

   Figure 4-1 shows an example of a button's properties in a Windows Forms project. Notice that the button's class name appears in the drop-down list at the top of the Properties window. In this figure, the fully qualified class name for a button is `System.Windows.Form.Button`.

   The class name is `Button`, and `System.Windows.Form` is the namespace used to access the `Button` class. You can use this information to look up the class documentation in the Visual Studio documentation.

   Web projects and Windows projects use different controls. They might go by the same name — in the case of a button, for example — but they're actually different classes. The fully qualified name for a Windows button is `System.Windows.Forms.Button`. You can read more about fully qualified namespaces in Chapter 2 of Book III.

Viewing a list of events

You can view a list of events in the Visual Basic Code Editor Navigation bar. To view a list of events via the Navigation bar, follow these steps:

1. **Repeat Steps 1 and 2 from the preceding step list.**

2. **Press F7 to open the Code Editor.**

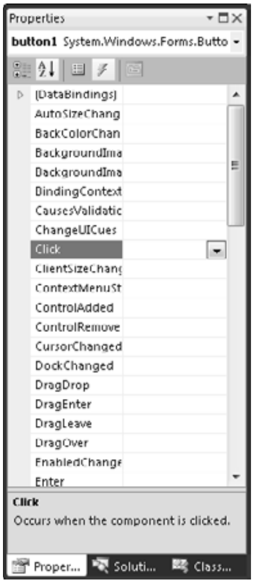
3. **Click the Class Name drop-down list on the Navigation bar.**

   You can find the Navigation bar at the top of the Code Editor.

4. **Select the control for which you wish to view events.**

5. **Click the Method Name drop-down list of the Navigation bar.**

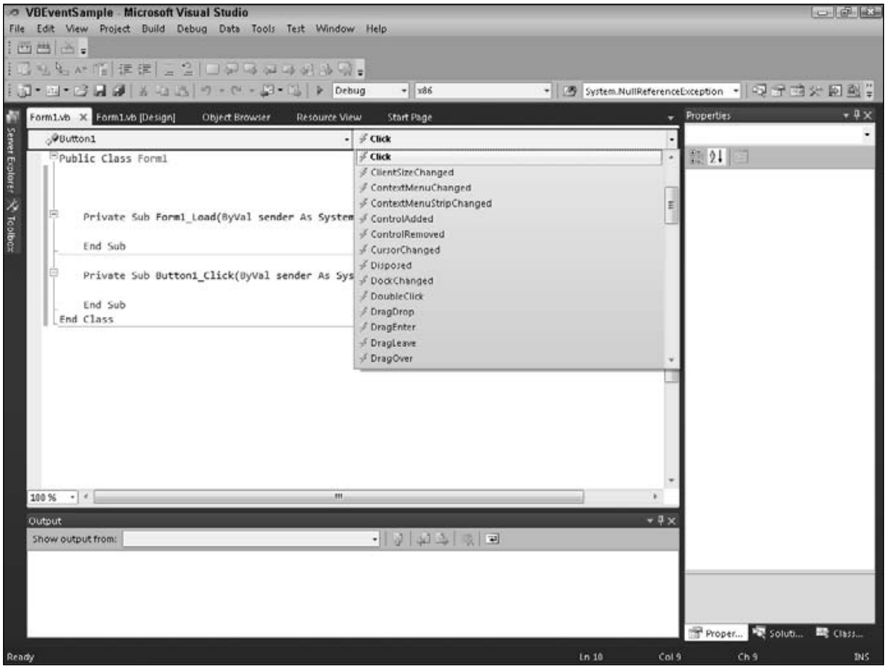
   A list of events appears, as shown in Figure 4-2. Notice the lightning bolt icons next to the event names.



Click

Occurs when the component is clicked.

Figure 4-1: A list of available events appears in the Properties window.



```
Public Class Form1


    Private Sub Form1_Load(ByVal sender As System

    End Sub

    Private Sub Button1_Click(ByVal sender As Sys

    End Sub
End Class
```

Figure 4-2: A list of events appears in the Code Editor Navigator bar.

Looking up event information

Having a list of event names is helpful, but it doesn't really tell you what the events do. You need to use the Visual Studio documentation to look up the class that creates the event. Before you can look up the event, you need to know the name of the class that creates the event. You can use the Properties window to get the fully qualified class name; refer to Figure 4-1.

Here's how to look up event information in the Visual Studio Help documentation:

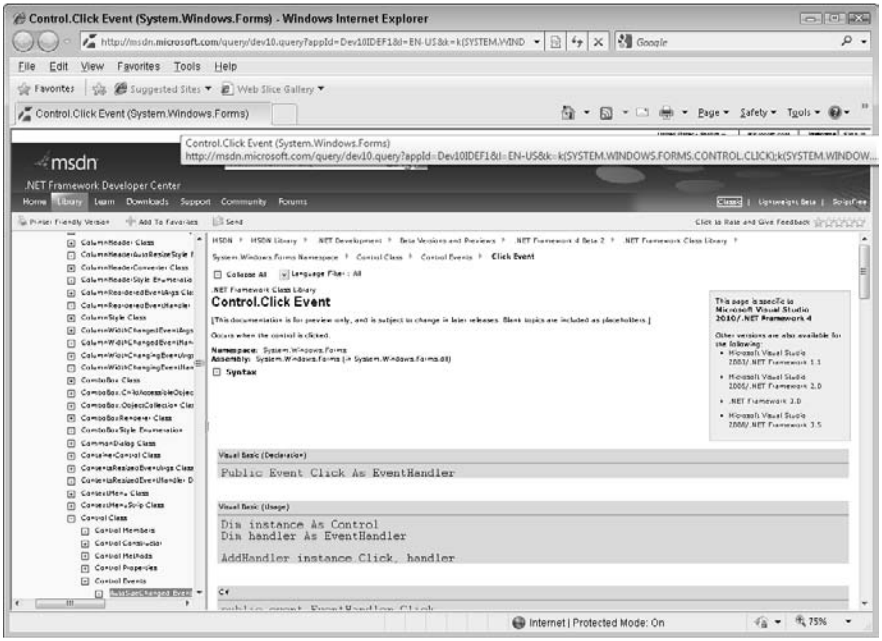
1. **Select the events icon in the Properties window.**
2. **Select an event in the Properties window.**
3. **Press F1.**

   The help for the event loads in your default web browser.

   A list of events for the class appears, as shown in Figure 4-3.

   A description appears for each event.
4. **Click an event to view details documentation about the event method.**

   You'll often find examples of how to write event handlers for the event method.



Control.Click Event (System.Windows.Forms) - Windows Internet Explorer

http://msdn.microsoft.com/query/dev10.query?appId=Dev10IDEF1&l=EN-US&k=k(SYSTEM.WINDOWS.FORMS.CONTROL.CLICK);k(SYSTEM.WINDOW...

MSDN › MSDN Library › .NET Development › Beta Versions and Previews › .NET Framework 4 Beta 2 › .NET Framework Class Library › System.Windows.Forms Namespace › Control Class › Control Events › Click Event

.NET Framework Class Library

Control.Click Event

[This documentation is for preview only, and is subject to change in later releases. Blank topics are included as placeholders.]

Occurs when the control is clicked.

Namespace: System.Windows.Forms
Assembly: System.Windows.Forms (in System.Windows.Forms.dll)

This page is specific to Microsoft Visual Studio 2010/.NET Framework 4

Other versions are also available for the following:
- Microsoft Visual Studio 2003/.NET Framework 1.1
- Microsoft Visual Studio 2005/.NET Framework 2.0
- .NET Framework 3.0
- Microsoft Visual Studio 2008/.NET Framework 3.5

Visual Basic (Declaration)

```
Public Event Click As EventHandler
```

Visual Basic (Usage)

```
Dim instance As Control
Dim handler As EventHandler

AddHandler instance.Click, handler
```

C#

File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Control.Click Event (System.Windows.Forms)
Page Safety Tools
msdn
.NET Framework Developer Center
Home Library Learn Downloads Support Community Forums
Classic | Lightweight Beta | ScriptFree
Printer Friendly Version
Add To Favorites
Send
Click to Rate and Give Feedback
Collapse All
Language Filter : All
Syntax
Internet | Protected Mode: On
75%

Figure 4-3: View a class' event details in the Visual Studio documentation.

Wiring up events and event handlers

For your code to execute when an event fires, the event must have some way to know your event handler exists. You create a link between the event and the event handler that essentially says, “When this event occurs, call me.”

You have several ways to create a link between an event and an event handler. Well, really, only one way: You have to register your event handler with the event method, which means writing code. But hardly anyone bothers to register their own event handlers because you can use Visual Studio to generate the code for you.

Creating a link between an event and an event handler is *wiring up the event handler.*

Here are the two steps to registering an event handler:

1. **Create the event handler.**

   You place the code that you want to execute when the event fires inside the event handler.

2. **Wire up the event handler to the event.**

Visual Studio takes care of both of these steps for you. You can either provide a handler for the default events or for the nondefault events, depending on your needs.

Default events

Every control or component designates a default event. The default event is usually the event most frequently fired for the control. For example, the `Click` event is the default event for a button on a Windows Form. And that makes sense when you consider that for buttons, the `Click` event is the most common event that is needed.

Visual Studio makes it really easy to register an event handler for a default event. All you have to do is double-click a control in the Windows or Web Forms Designers. For example, if you double-click a button control in the Windows Forms Designer using C#, Visual Studio generates the following code:

- **Event handler:**

  ```
  private void button1_Click(object sender, EventArgs e)
  {
  }
  ```

  The code block for the event handler appears in the code file for the Windows or Web Form.

Event handlers accept two arguments: a reference to the sender object that raised the event and any data associated with the event in the form of event arguments. Events that don't have any data use the `EventArgs` event data class provided by the .NET Framework. Any time you see a custom event data class, such as `MouseEventArgs e`, you know there's probably data available that your event handler can use. Otherwise, you can safely assume that the event doesn't have any data.

- **Wiring up the event handler to the event:**

```
this.button1.Click += new System.EventHandler(this.button1_Click);
```

The preceding code isn't readily visible to the programmer. In the case of Windows applications, the code appears in the partial class with the file extension `designer.cs` or `designer.vb`. In Web sites, the code is generated when the page is run.

`System.EventHandler` is a delegate data type provided by the .NET Framework for wiring up event handlers to events. `System.EventHandler` is not actually an event handler. Rather, `System.EventHandler` acts as the go-between for the event handler and the event. The event has no knowledge of the event handlers sitting out there waiting to handle the event. The event does have knowledge of the `System.EventHandler` delegate. `System.EventHandler` holds a reference to your event handler, which it passes along to the event.

Nondefault events

You can use Visual Studio to generate event handlers and wire them to events for nondefault events, too. To create event handlers for nondefault events, follow these steps:

1. **Open the Properties window for the control or component you want to handle an event for.**

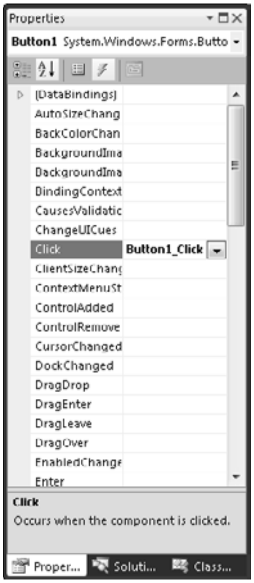
   See the preceding sections for an example of using the Properties window to view events.

2. **Locate the event you want to create an event handler for.**

3. **In the property grid, type the name of the event handler you want to create, as shown in Figure 4-4.**

   Alternatively, you can select an existing event handler from the drop-down list. You can also double-click the grid to create an event handler with a default name.

   The default name used by Visual Studio for event handlers is `variableName_Event`. For example, the `Click` event for `button1` is `button1_Click`.



Click Button1_Click

Click
Occurs when the component is clicked.

Figure 4-4: Use the Properties window to add event handlers for nondefault events.

Of course, an alternative approach to using the Properties window or a forms designer is to write your own event handler and wire the handler to the event. Earlier in the "Discovering Events" section, you can see the syntax for C#. What follows is the syntax for Visual Basic:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles Button1.Click

    End Sub
```

Getting a Life Cycle

If you've already started developing ASP.NET applications, perhaps you've been overwhelmed by the number and order of events that fired every time a user browsed to a Web page. However, after you take the time to understand the life cycle of ASP.NET applications and their resources — such as Web pages and Web services — figuring out when you want your code to execute can be a snap.

Each kind of application has its own life cycle events with which you should be familiar. ASP.NET applications, Windows applications, and mobile applications all have their own sets of controls and application life cycles. If you specialize in one of these kinds of applications, you need to understand what happens when a user fires up your application.

Understanding life cycles in Windows applications

Sometimes, waiting for an end user to click a button before your code executes is too late. For example, it's common to want to retrieve data from a database to populate a drop-down list. If you want the drop-down list populated before the user clicks a button, you need to find a different event to handle.

Startup events

When your application starts up, a series of events fire off. These events call the main form in your application, which in turn adds controls, such as the text boxes, menus, and buttons you dragged onto the form. Each control fires its own series of events as it goes about the business of drawing itself on-screen. The events of the application, main form, and controls describe a life cycle that you can use to insert your own code via event handlers.

When your application starts, the main form and controls in your application go through the following events:

- `HandleCreated`: Provisions resources from Windows for the control
- `Load`: Occurs before the form or control appears for the first time
- `Layout`: Occurs when child controls are added
- `Activated`: Occurs when the form is active
- `Shown`: Occurs the first time a form is displayed

The events just listed don't include every event that fires. Even more events exist. As you can see, you have a lot of opportunities to control what happens while forms and controls are appearing. For example, using the `Load` event to provision resources for controls on the form is fairly common. You might hit a database and populate a drop-down list during a form's `Load` event.

Shutdown events

Events aren't limited just to startup. Several events occur during shutdown, such as

- `FormClosing`: Occurs as the form is closing
- `FormClosed`: Occurs once the form has closed
- `Deactivate`: Occurs when the form loses focus and is no longer active

Events don't occur sequentially for parent and child controls. That is, you don't step through the form's events and then through the controls on the form. Rather, the events are nested. For example, imagine a form with a button and a text box. The `HandleCreated` event occurs three times, once for each control. The form's `HandleCreated` event occurs first, then the button's, and then the text box's. The form goes through its load, layout, activated, and shown events.

TIP

Forms are derived from the control class, so forms are controls like buttons and text boxes. Forms share a lot of events with other kinds of controls.

One final event, `ApplicationExit`, occurs before your application shuts down. You can use the `ApplicationExit` of the `Application` class to perform any cleanup tasks before your application shuts down.

Visual Basic application events

Visual Basic provides its own set of language-specific events for application startup and shutdown — `MyApplication`. To use `MyApplication`, follow these steps:

1. **Right-click your project in Solution Explorer.**
2. **Choose Properties from the shortcut menu.**

   The Project Designer appears.
3. **Click View Application Events on the Application tab.**

   The Visual Basic `MyApplication` class appears in the Code Editor.
4. **Add an event by using the Code Editor Navigation bar.**

Handling other events

Between startup and shutdown, any number of events might occur on forms and controls. Examples include `Paint`, `Lostfocus`, and `Leave`. Many seemingly simple activities have several events that make it happen. For example, consider a button click. Before the button's `Click` event occurs, the `MouseEnter`, `MouseDown`, and `MouseClick` events occur. After the click happens, the `MouseUp` event fires.

As you can see, you have plenty of opportunities to jump in and execute code without waiting on an end user to come along and interact with your program.

You may be asking yourself which events to use and in which order they should occur. Sometimes, you can figure it out just by reading a class' documentation, as described in the earlier section "Discovering events." Other options are to

- **Add event handlers for the events you're trying to discover.** You can use the debugger to step through the event handlers while code executes, write output to a file when the handler fires, or display text on a form.
- **Use reflection to discover events.** *Reflection* is a feature of the .NET Framework that essentially asks your code to look in a mirror and describe itself.

  The .NET Reflector application is a useful tool for exploring .NET assemblies and describing the data types included in that assembly. You can get the .NET Reflector application at the following address:

  ```
  http://www.red-gate.com/products/reflector
  ```

Understanding Web page life cycles

Say that you're sitting at your home in Denver, Colorado, where you browse to a Web page created with ASP.NET. Your request for a Web page must travel from your browser to a Web server in Kansas. The challenge in ASP.NET Web pages is getting the event from the client in Colorado to the server in Kansas. Thankfully, ASP.NET handles most of the details of making this work for you.

Nevertheless, you need to understand some concepts related to events in ASP.NET. At first blush, the event model seems similar to Windows applications. You double-click a control in the Web Forms Designer, and Visual Studio creates an event handler in your source code. In reality, it's more complicated than that. Many people get confused when their code doesn't execute as they expect it to.

Maintaining state in ASP.NET

Web pages are *stateless,* which means that ASP.NET forgets what it processes from page to page. Each request for a Web page is independent from previous requests. Of course, this isn't the experience that you have as an end user. You must understand a few concepts as a developer to keep this process transparent to your end users:

- **View state:** This is a hidden field in your Web page that stores data associated with a Web page's controls, such as the value in a text box.

  ASP.NET 2.0 introduced a view state feature called *control state,* which stores a limited set of a control's property data.
- **Postback:** *Postback* refers to an action that occurs when a Web page sends data back to the server for processing. Clicking a submit button on a form is an example of a postback.

  Read more about postbacks in the upcoming section "Handling postback events."

- **Render:** Web browsers don't understand the declarative syntax of ASP.NET or server-side code written with C# or Visual Basic. As a result, Web pages must be converted to — *rendered* as — HTML.

The ASP.NET life cycle starts when someone uses a Web browser to request a Web page. The ASP.NET life cycle includes three players:

- **Application:** The Web page request is sandwiched in a set of `Application` events that set up and tear down an environment to host and fulfill the client's request.
- **Page:** The Web page has its own life cycle where it builds itself and prepares the HTML that makes its way to the browser.
- **Controls:** Each server control has a life cycle that creates the control on the page.

Unlike Windows applications, Web pages live and die in one sitting. That is, the life cycles for the application, page, and controls last a matter of seconds. Each time a page is requested, the application, page, and controls are built, sent to the browser, and then discarded.

Examine these life cycles a little more closely to see what's going on. The very first request for a resource, such as a Web page for an ASP.NET Web site, results in the `ApplicationManager` being created for the entire Web site. The `ApplicationManager` sets up a hosting environment in which all subsequent requests are processed. This hosting environment is an *application domain.* Each ASP.NET Web site operates in its own exclusive application domain.

Handling page requests

Each request is handled within its own application inside the application domain. The application steps through a number of events that set up the execution environment to process the request, such as

- `BeginRequest`: Occurs before any other event during the processing of a request
- `AuthenticateRequest`: Establishes the identity of the user
- `AuthorizeRequest`: Ensures that the user has permission to process the request
- `PreRequestHandlerExecute`: Occurs before an event handler is called to process the request

At this point, the request is processed. For example, a request to a Web page calls the Web page's constructor, or a request to a Web service calls the Web service's constructor.

If the page has already been called, it might be stored in the ASP.NET cache. As a result, the request might be fulfilled from the cache rather than calling the resource each and every time a request is made.

An ASP.NET Web page has its own life cycle that it uses to fulfill the request. Here it is:

- `Page_PreInit`: This determines whether this is a new request or a postback request.
- `Page_Init`: Controls are available and accessible via each control's `UniqueID` property.
- `Page_Load`: Control properties are populated from view state and control state if the request is a postback.
- **Execute control events:** If the request is a postback, this controls which postback events, such as a button click, are fired.
- `Page_PreRender`: This is the last chance to make final changes to the page before its view state is saved.
- `Page_Render`: A view state is added to the page, and each control renders itself to HTML.
- `Page_Unload`: The page has been sent to the Web browser and is ready to be discarded.

After the page is discarded, the application wraps up the request processing by saving state data if required and updating the cache. The final event is the `EndRequest` event.

You'll interact with three kinds of events the most:

- **Postback events:** When users interact with your Web page by selecting items from drop-down lists and clicking buttons, it causes your Web page to post back to the server.
- **Page setup events:** You use page setup events, such as `Page_Load`, to perform tasks such as populate controls with data from databases respond to postbacks.
- **Application events:** You use application-level events to perform initial startup and shutdown tasks and handle errors.

Handling postback events

A postback occurs when a user interacts with your Web page and sends data back to the server. Postback events fire after the Web page's `Page_Load` event. Handling postback events is similar to handling events in Windows Forms.

Postback events, such as a button click, occur after the Web page's Page_Load event.

To create an event handler for a default event, double-click the control in the Web Forms Designer. You use the Properties window to set event handlers for nondefault events. See the section "Wiring up events and event handlers" for more details on creating event handlers with Visual Studio.

One thing that Visual Studio does that's different for ASP.NET Web page event handlers is that it adds the event handler to the Web server control's declaration in the ASP.NET page. For example, assume that you double-click a button in the Web Forms Designer. Visual Studio creates an event handler called Button1_Click. Visual Studio ties the event handler to the Web server's declaration by using the OnClick attribute, as shown in the following snippet:

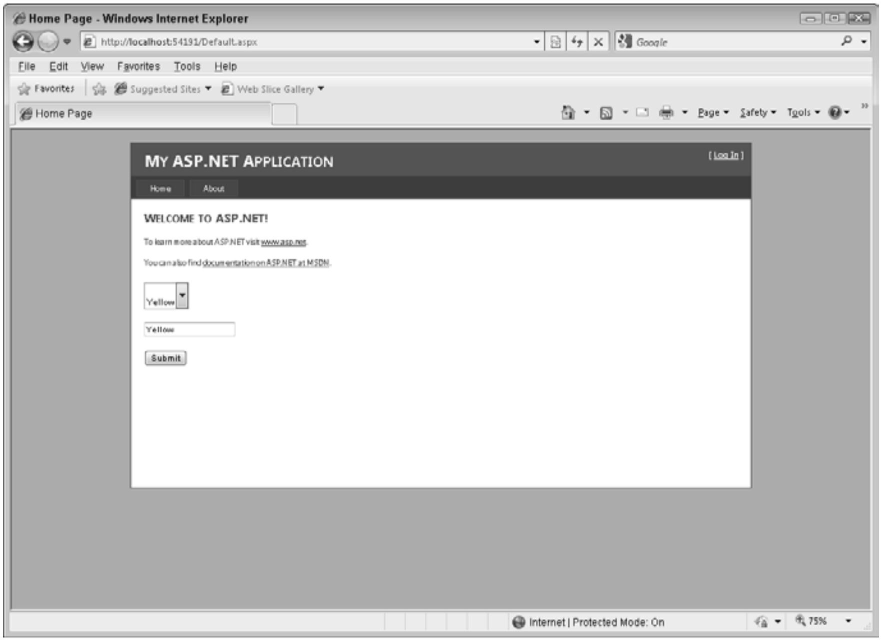
```
<asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Button" />
```

You don't need to designate an event handler for a button's Click event. The default behavior of a button that is clicked is to post back to the server. If you want code to execute in response to the Click event, you place it inside the button's Click event handler. Other postback events fire after the button's Click event handler.

You aren't restricted to processing just button clicks. You can respond to other events, such as when an item is selected from a drop-down list. The default event for a drop-down list is SelectedIndexChanged. This event allows you to capture the value a user selects from a drop-down list. For example, the following code uses the SelectedValue property of a drop-down list to place the selected value in a text box:

```
protected void DropDownList1_SelectedIndexChanged
    (object sender, EventArgs e)
{

   this.TextBox1.Text = this.DropDownList1.
    SelectedValue;
}
```

When the user selects a value from the drop-down list, it appears in the text box, as shown in Figure 4-5. The user must click the Submit button first in order to cause the page to post back. Alternatively, you can select the Enable AutoPostBack check box on the drop-down list's properties in the Form Designer.


Home Page - Windows Internet Explorer
http://localhost:54191/Default.aspx
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page
Safety
Tools
[Log In]
Home
About
Yellow
Submit
Internet | Protected Mode: On
75%

MY ASP.NET APPLICATION

WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

You can also find documentation on ASP.NET at MSDN.

Yellow

Figure 4-5: This event fires when the page posts back to the server.

An important concept in handling postbacks and postback events is to query the Web page's `IsPostback` property. The property returns `true` if the request is the result of a postback. It's common to check the `IsPostBack` property during the `Page_Load` event.

For example, say that you need to load items into a drop-down list. You create a procedure called `LoadItems`, as shown in the following code:

```
private void LoadItems()
{
  this.DropDownList1.Items.Add("Hello World");
  this.DropDownList1.Items.Add("Goodbye");
}
```

You assume that you can call `LoadItems()` in the `Page_Load` event. Unfortunately, when you do that, your `LoadItems()` procedure is called every time the page posts back, so your drop-down list is loaded every time. The `IsPostBack` property allows you to test whether the page is posting back. The `IsPostBack` is a Boolean property, which means that it returns one of two values:

- `True`: If you want your code to execute every time after the page's first load, use this statement.
  - **C#:**

    ```
    If (Page.IsPostback == true)
    ```

  - **Visual Basic:**

    ```
    If Page.IsPostBack = True Then
    End If
    ```

- `False`: If you want your code to run only the first time the user accesses it, you test the `IsPostBack` property for `false`.
  - *Alternatively, you can use the C#* `not` *operator* `!`*, as shown here:*

    ```
    If(!Page.IsPostBack)
    ```

  - *Using the Visual Basic keyword* `Not` *looks like this:*

    ```
    If Not Page.IsPostBack Then
    End If
    ```

You just include your code inside the `If` clause, as the following code shows:

```
protected void Page_Load(object sender, EventArgs e)
{
  if (Page.IsPostBack == false)
     LoadItems();
}
```

TIP

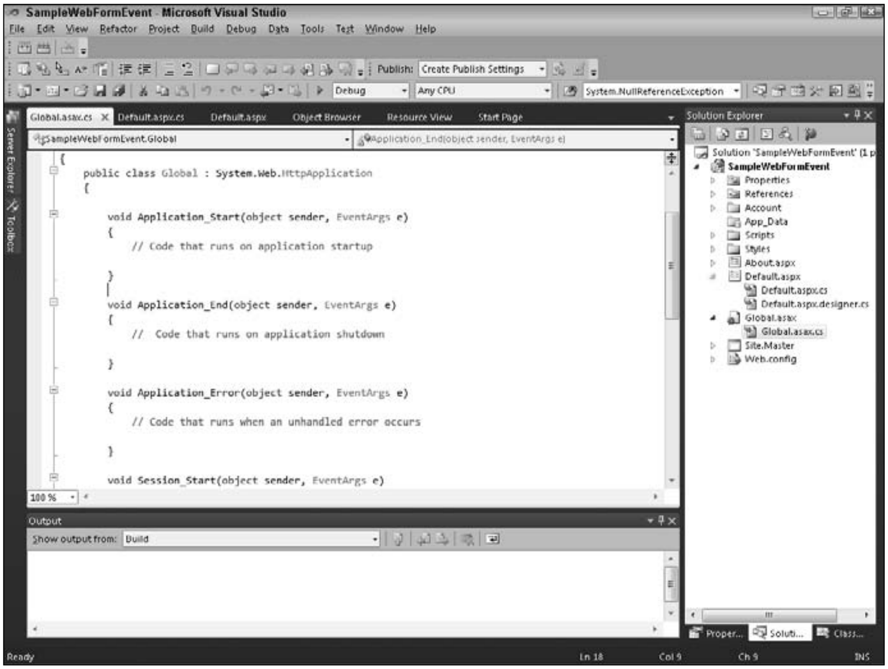
Many controls have an `AutoPostBack` property that you can enable. When this property is enabled, the control causes a postback without requiring the user to click a button.

Handling application events

Sometimes, you need to capture an event that's related to the application. For example, you may want to log the first time when an application starts or when errors occur. ASP.NET provides application-level events for which you can write event handlers. You write event handlers in the global application class file. Visual Studio 2010 adds the global application class file named `global.asax` for you, as shown in Figure 4-6.

You can add several application-level events to the `global.asax` file, including the following:

- `Application_Start`: Fires only the very first time a Web site is accessed
- `Application_End`: Fires only once before the entire Web site is shut down
- `Application_Error`: Fires any time an unhandled exception occurs



```
{
    public class Global : System.Web.HttpApplication
    {

        void Application_Start(object sender, EventArgs e)
        {
            // Code that runs on application startup

        }

        void Application_End(object sender, EventArgs e)
        {
            //  Code that runs on application shutdown

        }

        void Application_Error(object sender, EventArgs e)
        {
            // Code that runs when an unhandled error occurs

        }

        void Session_Start(object sender, EventArgs e)
```

Figure 4-6: Place application-level events in this file.

Global.asax derives from the HttpApplication class. As a result, Global.asax supports all the events of the HttpApplication class. Application_Start and Application_End are special events that aren't part of the HttpApplication class. ASP.NET knows to fire these two events at the very beginning and the very end of a Web site's life cycle.

Application_Error is your last opportunity to catch unhandled exceptions in your ASP.NET applications. It's common practice to use Application_Error as the event handler to place code that you want executed every single time an execution is thrown.

For example, assume that you have a Try/Catch block in your Web page. In your Catch clause, you simply use the throw statement, as shown here:

```
Try
 'some code here
Catch ex As Exception
 'attempt to handle the exception
 Throw
End Try
```

The `throw` statement passes the exception to `Application_Error` event handler where you have the chance to log the exception. `Application_Error` allows you to pass your user to a friendly error-message page, if appropriate.

Be sure to use `Server.ClearError()` to clear the exception before you pass the caller from `Application_Error` to another page in your application.

`Global.asax` allows you to place code in the `Session_Start` and `Session_End` event handlers. `Session_Start` is a very reliable event that you can use to log that a new session has started. You can use `Session_End` to clean up resources.

Many additional events are available to you in the application life cycle. Unfortunately, not all these events fire consistently. The only event that's guaranteed to fire, besides the three application events mentioned earlier, is `Application_EndRequest`.

Chapter 5: Getting Productive with Visual Studio 2010

In This Chapter

- **Analyzing your code with FxCop**
- **Digging into the new IntelliSense features**
- **Creating XML documentation from code comments**
- **Refactoring C# code**

The promise of the latest suite of Visual Studio products is personal productivity coupled with extended functionality. Features (such as Language Integrated Query [LINQ]) make writing code substantially faster, and other features (such as Asynchronous JavaScript and XML [AJAX]) make your code substantially more efficient and flexible. However, the most important productivity aids are the ones you use every day and in every project, such as IntelliSense. The features I cover in this chapter show you how to take your productivity with Visual Studio 2010 to the next level.

This chapter shows you how to use Visual Studio 2010 Professional to analyze your code with FxCop. You can also read about the code snippets and refactoring features. If you're new to object-oriented programming (OOP), Visual Studio 2010 has several IntelliSense features that make light work of turning your code into objects.

Sending Your Code to Boot Camp

Have you ever wondered how some developers know what to name their variables and custom data types? Does some secret society exist in which all the in-the-know developers get memos on when to use a field instead of a property? Actually, there is — sort of.

Attending the naming convention

The .NET Framework software development kit (SDK) contains the Design Guidelines for Class Library Developers. Even though these design guidelines are indeed meant for class library developers, you can take advantage of the recommendations in the guidelines when writing your own code. Some of the topics that the guidelines cover include

- Naming conventions
- Using data types and class members
- Handling errors and raising exceptions

TIP

If nothing else, the naming conventions outlined in the design guidelines are helpful. Using naming conventions makes it very easy to quickly identify the purpose of a block of code just by looking at how it's named. It also reinforces the concepts in OOP because you associate certain naming conventions with specific OOP constructs.

The following two case standards are used the most in naming conventions:

- **Camel casing:** In a concatenated word, the first letter is lowercase, and the first letter of the second half is uppercase, as in dogBark. If you have more than two words, all the following words also initial capped: for example, dogBarkLoud.
- **Pascal casing:** In a concatenated word, the first letters of both the first and second words are uppercase, as in DogBark.

Table 5-1 shows some of the naming guidelines that use camel and Pascal casing.

Table 5-1 Naming Guidelines

Case	*Identifier*	*Example*
Camel	Fields	`customerName`
	Local variables	`employeeID`
	Parameters	`updateSql`
Pascal	Class	`Customer`
	Method	`ValidateAddress()`
	Property	`CompanyName`

Other examples of conventions include prefixing the letter `I` to interfaces, such as `IEnumerable`; and appending the word `Exception` to derived exceptions, such as `NullReferenceException`.

You should use the keywords `Me` (in Visual Basic) and `this` (in C#) to distinguish between fields and local variables. For example, the following code uses a local string variable named `text` and a field named `greetingName` in a method:

```
Public Function DisplayMessage() As String
  Dim text As String
  If String.IsNullOrEmpty(Me.greetingName) Then
     text = SayHello()
  Else
     text = SayHelloName()
  End If
    Return text
  End Function
```

One convention not covered in the design guidelines is how to name controls. Many developers still use Hungarian notation-style prefixes in front of their controls — such as `txt` for *text box* or `btn` for *button.* For details on Hungarian notation, see these references:

`www.fmsinc.com/dotnet/Analyzer/Rules/Hungarian.htm`

`www.byteshift.de/msg/hungarian-notation-doug-klunder`

If you want to use Hungarian notation, you can, as long as you're consistent. Some developers like to use a single prefix for all controls, such as `UI` for *user interface.* Some developers have started to use `UX` for *user experience.* Either way, all your controls are grouped, as shown in Figure 5-1.

Charles Simonyi created the concept of using prefixes to identify variables. He's originally from Hungary, which is why this style of notation is called Hungarian notation. You can find a list of Hungarian naming conventions for Visual Basic in Microsoft's knowledge base at

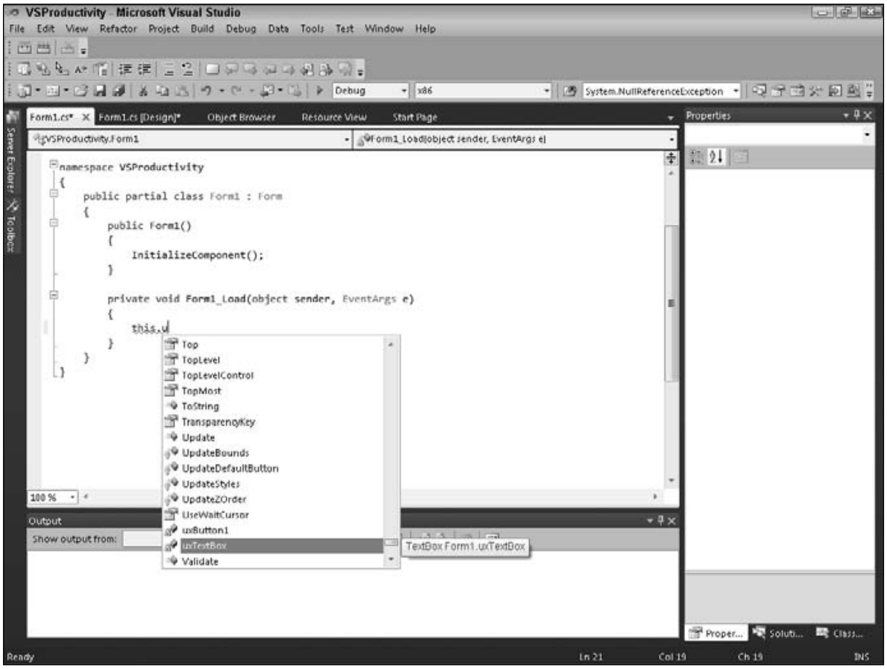
`http://support.microsoft.com/kb/q173738`

Search for the topic "Class Library Design Guidelines" in the Visual Studio documentation to access the complete guidelines.

Calling all cars! FxCop to the rescue!

You're likely thinking that you don't have time to read a bunch of rules. You have deadlines. You get paid to bang out code, not read rulebooks. And you certainly don't have the discipline it takes to review your code for compliance to rules.

What you need is a *cop* — someone who (or something that) can police your code and tell you when you violate the rules. *FxCop* is a code analysis tool that analyzes your code for conformance to the .NET Design Guidelines for Class Library Developers.



```
namespace VSProductivity
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            this.u
        }
    }
}
```

Figure 5-1: Using a single prefix on all user interface controls groups them.

You can download the software for free from the MSDN Web site at `www.microsoft.com/downloads/en/default.aspx`. The Microsoft Downloads site loads in your browser. From here, enter **FxCop** in the search box and choose Microsoft FxCop 1.36 from the list of search results.

The latest version targets version 2.0 of the .NET Framework, but it will do an analysis of any version of the Framework — it just needs version 2.0 of the Framework to operate.

Downloading FxCop with Vista

For those of you working with Vista, you might notice that you can't even download FxCop. The first step is to set the Web site as one of your trusted Web sites, using the following steps.

1. **Double click the Internet icon at the bottom of Internet Explorer.**

   You see the Internet Security dialog box.

2. **Highlight the Trusted Zones entry.**
3. **Click Sites.**

   You see the Trusted Sites dialog box. The URL of the site appears in the Add This Web Site to the Zone field.

4. **Clear the Require Server Verification (https:) For All the Sites in This Zone option.**
5. **Click Add.**

   Internet Explorer adds the Web site to the list of trusted Web sites.
6. **Click Close and then click OK.**

   You see the Internet icon change to a Trusted Sites icon (with a green check mark).

After you enable the download, you'll probably notice that Vista prevents you from saving the file. To overcome this problem, you must perform this set of unlikely steps:

1. **Choose Start⇨Programs⇨Windows Mail to open Windows Mail.**

   You see an initial display that might include a wizard for configuring Windows Mail. If you see the wizard, click Cancel and then click Yes to cancel the wizard.
2. **Choose Tools⇨Options to display the Options dialog box. Choose the security tab.**

   You see a list of security options.
3. **Clear the Do Not Allow Attachments to be Saved or Opened That Could Potentially Be a Virus option.**
4. **Click OK.**
5. **Close Windows Mail.**

At this point, you can finally download and save the file. Log out of Vista and then back in to make the changes permanent. To download the file, right-click the link and choose Save Target As from the context menu. You'll see a dialog box for saving the file on your system.

Visual Studio Team System (VSTS) has integrated support for code analysis using FxCop. For more on Visual Studio Team System, turn to Book VII, Chapter 2.

Analyzing an assembly

FxCop analyzes compiled assemblies. You need to build your project at least once before FxCop can analyze your code. You can build your project by using the Build menu. See Book VI, Chapter 1 for an in-depth discussion of builds. To use FxCop to analyze an assembly, follow these steps:

1. **Open FxCop.**

   Access FxCop from the All Programs menu from your Start button.

2. **From the Project menu, choose Add Targets.**

   The Open window appears.

   The assemblies you want to analyze are called *targets* in FxCop.

3. **Browse to the location of your assemblies.**

   Note that assemblies are located in the bin folder in your solution's directory.

4. **Select the assembly you want to analyze and then click the Open button.**

   Note: You may need to locate `System.Core.dll` for version 3.5 of the .NET Framework. If this is the case, navigate to

   ```
   C:\Windows\assembly\GAC_MSIL\System.Core\3.5.0.0__b77a5c561934e089
   ```

   and load `System.Core.dll`.

   The assembly appears in FxCop. (Press the Shift key to select multiple assemblies.)

5. **Choose Project➪Analyze.**

   The FxCop engine analyzes your assemblies. The results appear in FxCop.

6. **Browse through the list of warnings and select one to view its details.**

TIP

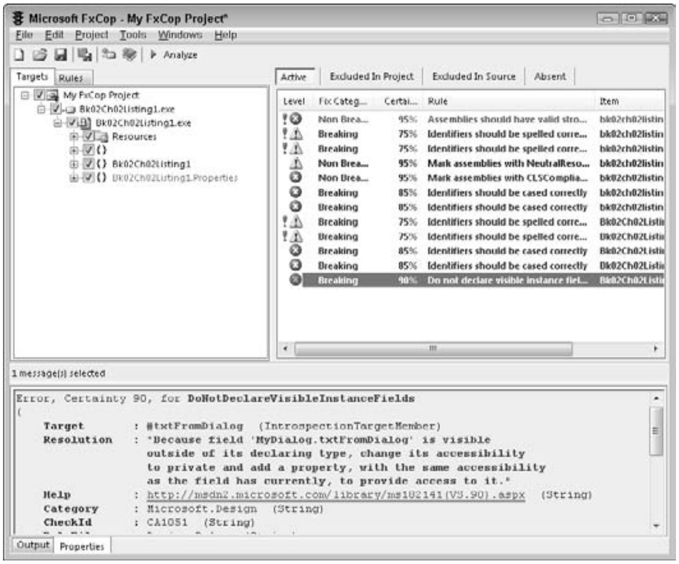
The first time you use FxCop, you'll likely be overwhelmed by the volume of messages. You might like to sort the list by the Rules column while you step through and examine the messages.

When you click a message in FxCop, a description of the message appears. The description provides you with

- Member under inspection
- A link to the source code
- Problem description and suggestion resolutions
- A link to an online help description of the problem and resolution

Dealing with errors

Figure 5-2 shows an error triggered by declaring a field that is visible outside of its declaring type. This is described as a "Breaking" change because changing the accessibility of this field may cause code that references it not to build.



Level	Fix Categ...	Certai...	Rule	Item
! ⊗	Non Brea...	95%	Assemblies should have valid stro...	bk02ch02listin
! ⚠	Breaking	75%	Identifiers should be spelled corre...	bk02ch02listin
! ⚠	Breaking	75%	Identifiers should be spelled corre...	bk02ch02listin
⚠	Non Brea...	95%	Mark assemblies with NeutralReso...	bk02ch02listin
⊗	Non Brea...	95%	Mark assemblies with CLSComplia...	bk02ch02listin
⊗	Breaking	85%	Identifiers should be cased correctly	bk02ch02listin
⊗	Breaking	85%	Identifiers should be cased correctly	bk02ch02listin
! ⚠	Breaking	75%	Identifiers should be spelled corre...	Bk02Ch02Listi
! ⚠	Breaking	75%	Identifiers should be spelled corre...	Bk02Ch02Listi
⊗	Breaking	85%	Identifiers should be cased correctly	Bk02Ch02Listi
⊗	Breaking	85%	Identifiers should be cased correctly	Bk02Ch02Listi
⊗	Breaking	90%	Do not declare visible instance fiel...	Bk02Ch02Listi

1 message(s) selected

```
Error, Certainty 90, for DoNotDeclareVisibleInstanceFields
(
    Target        : #txtFromDialog  (IntrospectionTargetMember)
    Resolution    : "Because field 'MyDialog.txtFromDialog' is visible
                    outside of its declaring type, change its accessibility
                    to private and add a property, with the same accessibility
                    as the field has currently, to provide access to it."
    Help          : http://msdn2.microsoft.com/library/ms182141(VS.90).aspx  (String)
    Category      : Microsoft.Design  (String)
    CheckId       : CA1051  (String)
```

Figure 5-2: FxCop provides detailed descriptions of the problems discovered and suggests resolutions.

When you click the source code hyperlink, the offending code opens in Visual Studio. (Vista might show one or more error messages, depending on how you set the User Access Control [UAC] feature.) Sure enough, the field `txtFromDialog` is declared public:

```
public System.Windows.Forms.TextBox txtFromDialog;
```

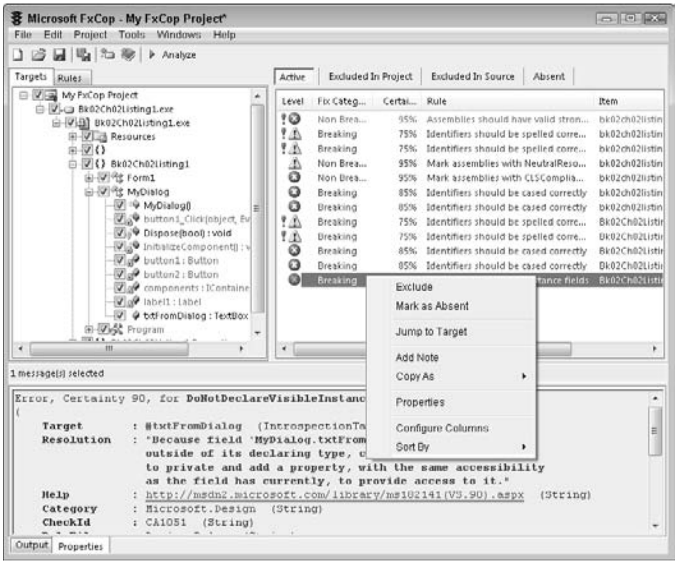
By changing the code to the following, the warning goes away:

```
private System.Windows.Forms.TextBox txtFromDialog;
public System.Windows.Forms.TextBox TextFromDialog
        {
            get
            {
                return (this.txtFromDialog);
            }
        }
```

If you come across a certain rule that you don't care about enforcing, you can choose to exclude the rule. To exclude the rule, right-click the message and choose Exclude, as shown in Figure 5-3.

TIP

If you don't like the rules included with FxCop, you can create your own. See the documentation for FxCop for more information.



Level	Fix Categ...	Certai...	Rule	Item
	Non Brea...	95%	Assemblies should have valid stron...	bk02ch02listin
	Breaking	75%	Identifiers should be spelled corre...	bk02ch02listin
	Breaking	75%	Identifiers should be spelled corre...	bk02ch02listin
	Non Brea...	95%	Mark assemblies with NeutralReso...	bk02ch02listin
	Non Brea...	95%	Mark assemblies with CLSComplia...	bk02ch02listin
	Breaking	85%	Identifiers should be cased correctly	bk02ch02listin
	Breaking	85%	Identifiers should be cased correctly	bk02ch02listin
	Breaking	75%	Identifiers should be spelled corre...	Bk02Ch02Listi
	Breaking	75%	Identifiers should be spelled corre...	Bk02Ch02Listi
	Breaking	85%	Identifiers should be cased correctly	Bk02Ch02Listi
	Breaking	85%	Identifiers should be cased correctly	Bk02Ch02Listi
	Breaking		tance fields	Bk02Ch02Listi

1 message(s) selected

```
Error, Certainty 90, for DoNotDeclareVisibleInstanc
(
    Target        : @txtFromDialog  (IntrospectionTa
    Resolution    : "Because field 'MyDialog.txtFrom
                    outside of its declaring type, c
                    to private and add a property, with the same accessibility
                    as the field has currently, to provide access to it."
    Help          : http://msdn2.microsoft.com/library/ms182141(VS.90).aspx  (String)
    Category      : Microsoft.Design  (String)
    CheckId       : CA1051  (String)
```

Figure 5-3: Exclude rules that aren't relevant for your situation.

Setting up FxCop as an external tool

FxCop features a command line version that you can include in build scripts or integrate into Visual Studio as an external tool. To integrate FxCop as an external tool in Visual Studio, just follow these steps:

1. **Choose Tools➪External Tools.**

   The External Tools window appears.

2. **Click the Add button if this isn't the first tool added to the menu.**
3. **Type** Run FxCop **in the Title field.**
4. **Browse to the FxCop command line executable for the Command field.**

   The default value for Windows XP is

   ```
   C:\Program Files\Microsoft FxCop 1.36 \FxCopCmd.exe
   ```

   The default value for Windows Vista (64-bit) is

   ```
   C:\Program Files (x86)\Microsoft FxCop 1.36\
       FxCopCmd.exe for Windows Vista
   ```

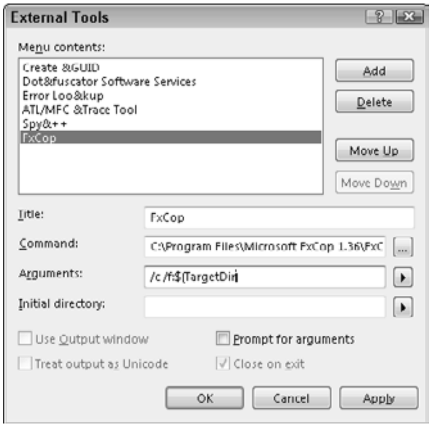
   Note: The default drive may not be C: if you install FxCop on a different drive.

5. **Type arguments that you normally use on the command line.**

   For example, to analyze files using the solution's build directory, use the argument variable `$(TargetDir)`. You can click the arrow next to the Arguments field to select an argument variable, as shown in Figure 5-4.

6. **Enable the Use Output Window and Prompt for Arguments check boxes and then click OK.**

   A new menu entry, Run FxCop, appears on the Tools menu.



External Tools

Menu contents:

Create &GUID
Dot&fuscator Software Services
Error Loo&kup
ATL/MFC &Trace Tool
Spy&++
FxCop

Title: FxCop

Command: C:\Program Files\Microsoft FxCop 1.36\FxC

Arguments: /c /f:$(TargetDir

Initial directory:

□ Use Output window □ Prompt for arguments

□ Treat output as Unicode ☑ Close on exit

Figure 5-4: Add FxCop as an internal tool in Visual Studio.

It's Not Your Father's IntelliSense Any More

For years, developers have been using homegrown tools and utilities, such as CodeSmith, to generate repetitive code. The IntelliSense features in Visual Studio 2010 do way more than generate repetitive code. IntelliSense supports coding styles such as test-driven development (TDD) and OOP. If your style is copy and paste, IntelliSense has a tool for you, too.

Using code snippets

A major goal for almost all development projects is to increase code reuse. IntelliSense code snippets are a perfect example of reuse in its simplest form.

Code snippets are like copy and paste on steroids. When using shortcut keywords or a menu, you can browse a library containing hundreds of code samples that you can insert into your code. Best of all, the code samples work like templates that highlight the fields where you need to insert code specific to your application.

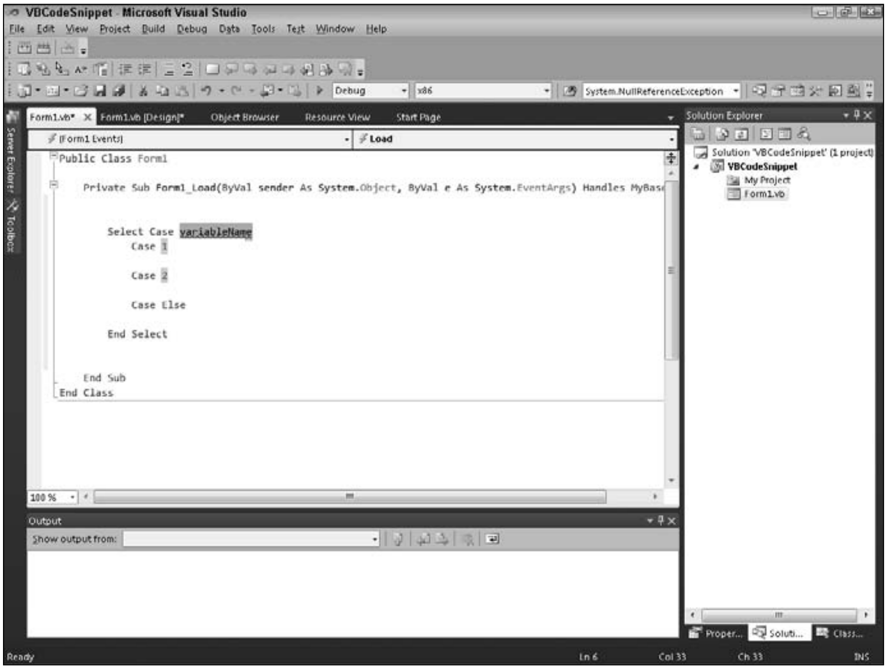
For example, if you always forget the syntax for `Select` statements, you can use a code snippet to refresh your memory. To insert a `Select` code snippet in Visual Basic, simply follow these steps:

1. **Position your cursor where you want the code inserted.**
2. **Type** select **and press the Tab key.**

   A code snippet appears in the code editor, as shown in Figure 5-5.

   The blocks highlighted in green indicate where you place your values.
3. **Type your values.**

   Press Tab to maneuver between the highlighted blocks.



```
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase


        Select Case variableName
            Case 1

            Case 2

            Case Else

        End Select


    End Sub
End Class
```

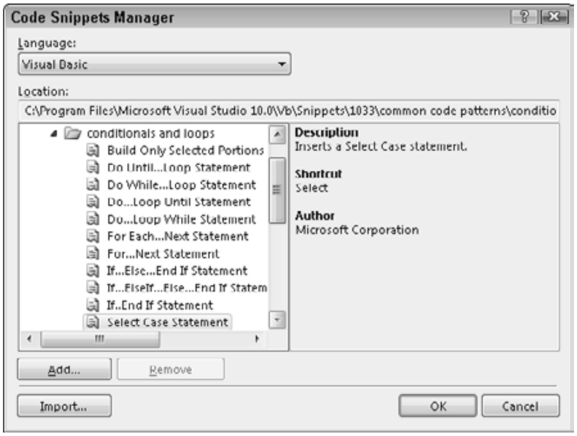
Figure 5-5: The code snippet appears as a template.

In the preceding example, you use the code snippet's shortcut to access the snippet. If you don't know the shortcut, you can select a snippet using the following options:

- Choose Edit➪IntelliSense➪Insert Snippet.
- Right-click the Code Editor and choose Insert Snippet from the shortcut menu that appears.
- In Visual Basic, type a question mark (?) and then press the Tab key.

Position your cursor where you want the snippet to appear in the Code Editor. If you just want to browse snippets without selecting a snippet, use the Code Snippets Manager. You access the manager by choosing Tools➪Code Snippets Manager.

The Code Snippets Manager allows you to access snippets for many programming languages in one screen. The manager displays a snippet's location, description, and shortcut. *Snippet files* use XML syntax and end in the file extension `.snippet`. You can add snippet files you create or download from the Web via the Code Snippets Manager, as shown in Figure 5-6.



Code Snippets Manager

Language: Visual Basic

Location: C:\Program Files\Microsoft Visual Studio 10.0\Vb\Snippets\1033\common code patterns\conditio

conditionals and loops
- Build Only Selected Portions
- Do Until...Loop Statement
- Do While...Loop Statement
- Do...Loop Until Statement
- Do...Loop While Statement
- For Each...Next Statement
- For...Next Statement
- If...Else...End If Statement
- If...ElseIf...Else...End If Statem
- If..End If Statement
- Select Case Statement

Description
Inserts a Select Case statement.

Shortcut
Select

Author
Microsoft Corporation

Figure 5-6: Use the Code Snippets Manager to browse and add code snippets.

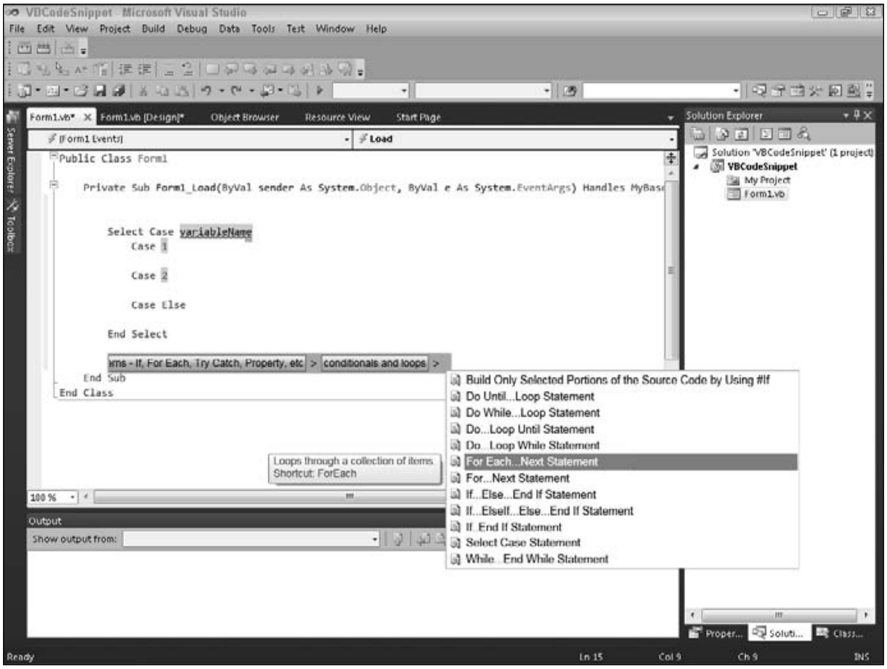
When you access code snippets via a menu, you use the Code Snippets Picker. Snippets are organized into folders in Visual Basic. C# provides a list of snippets. To access a snippet via the Code Snippets Picker, do the following:

1. **Position your cursor where you want to insert the code snippet.**
2. **Right-click the Code Editor.**
3. **From the shortcut menu that appears, choose Insert Snippets.**
4. **From the Code Snippet Picker that appears, select a snippet from its list.**

   The snippet appears in the Code Editor.

   In Visual Basic, use the Tab key to navigate the folders until you find the snippet you need.

   Hover your mouse over the snippet to see the snippet's shortcut, for future reference, as shown in Figure 5-7.


VBCodeSnippet - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Test Window Help
Form1.vb* Form1.vb [Design]* Object Browser Resource View Start Page
(Form1 Events) Load
Solution Explorer
Solution 'VBCodeSnippet' (1 project)
VBCodeSnippet
My Project
Form1.vb
Server Explorer Toolbox
100 %
Output
Show output from:
Proper... Soluti... Class...
Ready Ln 15 Col 9 Ch 9 INS

```
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase


        Select Case variableName
            Case 1

            Case 2

            Case Else

        End Select

        rns - If, For Each, Try Catch, Property, etc > conditionals and loops >
    End Sub
End Class
```

Build Only Selected Portions of the Source Code by Using #If
Do Until...Loop Statement
Do While...Loop Statement
Do...Loop Until Statement
Do...Loop While Statement
For Each...Next Statement
For...Next Statement
If...Else...End If Statement
If...ElseIf...Else...End If Statement
If...End If Statement
Select Case Statement
While...End While Statement

Loops through a collection of items
Shortcut: ForEach

Figure 5-7: Use the Code Snippet Picker to find a snippet and view its shortcut.

Visual Basic provides a broader range of snippets for everything from basic language constructs to accessing resources in Windows. C# snippets are limited to flow control, as well as data type and member declaration snippets. C# has a folder of refactoring snippets that you can see demonstrated in the upcoming section "Factoring in the Refactoring."

Stub it out with method stubs

One of the complaints about development environments like Visual Studio is that they require you to define everything from the bottom up. Some people complain that this style requires you to dig into the minutiae before you have a chance to see the big picture of how your program should be laid out. An IntelliSense feature that allows you to generate method stubs from a method call makes it easy to stay focused.

A *code stub* is a placeholder for an unimplemented piece of code.

Generating method stubs is common in TDD. Adherents to TDD believe you should write tests first. You can use IntelliSense to generate method stubs from unit tests.

In essence, the Generate Method Stub feature allows you to use a method call before you write it. You invoke the Generate Method Stub feature to generate the method's declaration from the method call.

To create a new method stub from a method call, perform the following steps:

1. **Create a method call in your code.**

   Be sure to specify any parameters. If you want the method stub to return a value, assign the method call to a variable of the correct data type.

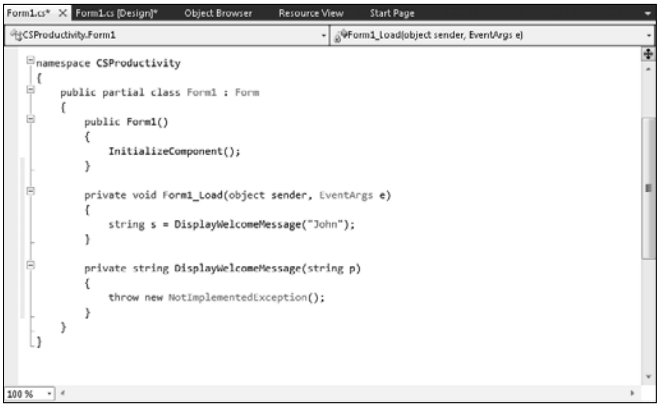
   For example, the following method call accepts a `string` parameter and returns a string:

   ```
   string s = DisplayWelcomeMessage("John");
   ```

2. **Right-click the method call and choose Generate Method Stub from the shortcut menu.**

   IntelliSense generates the method stub, as shown in Figure 5-8.

   Note that the method body throws an exception if the method is called.



```
namespace CSProductivity
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            string s = DisplayWelcomeMessage("John");
        }

        private string DisplayWelcomeMessage(string p)
        {
            throw new NotImplementedException();
        }
    }
}
```

Figure 5-8: IntelliSense can generate a method stub from a method call.

Adding using statements

If you're like some developers, you just start typing away in the Code Editor and forget to reference all the namespaces you need with a `using` statement.

IntelliSense has a feature that adds `using` statements for data types that aren't fully qualified. This feature can be helpful when you access a type without adding the `using` statement or when you copy code from another resource, such as a sample on the Web.

To add a `using` statement, do the following:

1. **Add an unbound data type (one that is not directly bound to a data source) to your code.**

   For example, the following code uses the `SqlCommand` without fully qualifying the namespace:

   ```
   SqlCommand cmd = new SqlCommand();
   ```

2. **Position your cursor on the last letter of the data type.**
3. **Hover your mouse over the last letter of the data type.**

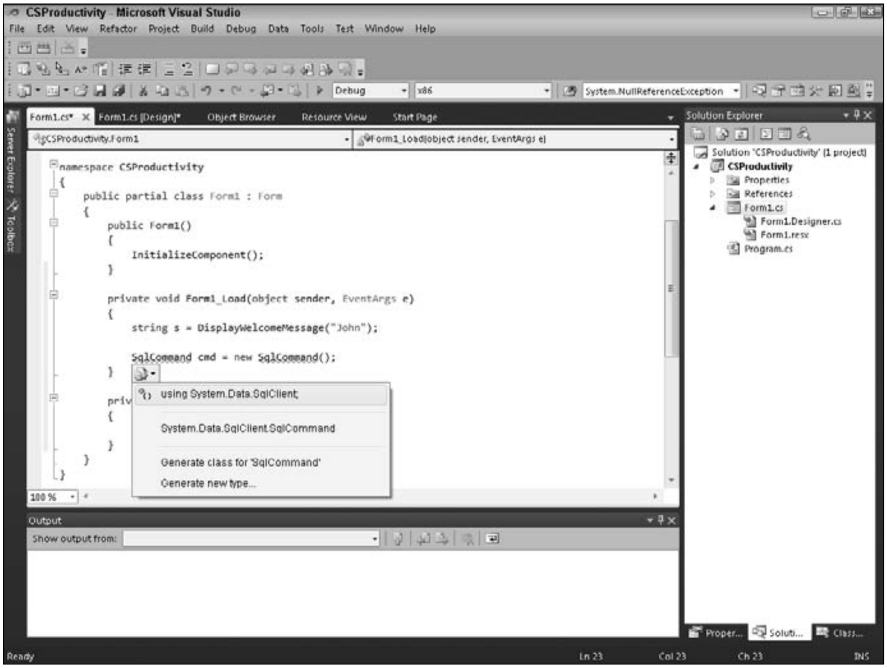
   A Smart Tag appears.

4. **Click the Smart Tag and select the `using` statement, as shown in Figure 5-9.**

   Visual Studio adds the `using` statement to the top of the source code file.

   For example, the following statement is added for the `SqlCommand` type:

   ```
   using System.Data.SqlClient;
   ```



```
namespace CSProductivity
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            string s = DisplayWelcomeMessage("John");

            SqlCommand cmd = new SqlCommand();
        }

        priv
        {

        }
    }
}
```

CSProductivity - Microsoft Visual Studio
File Edit View Refactor Project Build Debug Data Tools Test Window Help
Debug
x86
System.NullReferenceException
Form1.cs* Form1.cs [Design]* Object Browser Resource View Start Page
CSProductivity.Form1
Form1_Load(object sender, EventArgs e)
using System.Data.SqlClient;
System.Data.SqlClient.SqlCommand
Generate class for 'SqlCommand'
Generate new type...
100 %
Output
Show output from:
Solution Explorer
Solution 'CSProductivity' (1 project)
CSProductivity
Properties
References
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Proper... Soluti... Class...
Server Explorer
Toolbox
Ready
Ln 23
Col 23
Ch 23
INS

Figure 5-9: Add a `using` statement for an unbound data type.

TIP

This feature works for only C# using statements. You can't use it with Visual Basic Imports statements.

Objectifying your code

IntelliSense provides several features that make quick work of common, object-oriented tasks. In OOP, a number of abstract elements are defined. Developers are responsible for writing the code that implements the abstract elements. The concrete implementations are said to *inherit,* or *implement,* the abstract elements.

TIP

Abstract classes and interfaces are two abstract elements used in OOP. You inherit from an abstract class and implement an interface.

The process of inheriting or implementing from abstract elements requires a lot of typing. For example, assume that you want to inherit from an abstract class that has ten members. You have to re-create all ten members in your derived class.

IntelliSense provides features that copy abstract methods to your concrete classes. You can access these features from a shortcut menu in the Code Editor or Class Designer.

Implementing abstract classes

An *abstract class* defines a class that is used for inheritance only. You can't create an instance of an abstract class. Rather, you must create a new class that inherits from or implements the abstract class. The class you implement is a *concrete class* because you can create an instance of the class.

IntelliSense provides a feature that makes quick work of inheriting from an abstract base class. Here's how to inherit from an abstract class:

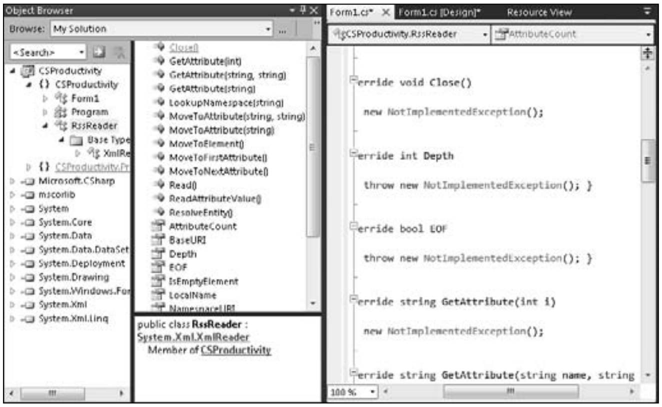
1. **Create a new concrete class.**
2. **Type a colon (:) after the class name declaration.**

   For example, assume that you want to create a new class called RssReader that inherits from the abstract class XmlReader. Type the following:

   ```
   public class RssReader: System.Xml.XmlReader
   ```
3. **Right-click the abstract class and choose Implement Abstract Class from the shortcut menu.**

   IntelliSense creates method stubs for all the members of the abstract class.

Figure 5-10 shows the concrete class `RssReader` in Object Browser. The class inherits from `XmlReader` and has all the same methods.



public class **RssReader** :
System.Xml.XmlReader
Member of CSProductivity

```
erride void Close()

  new NotImplementedException();

erride int Depth

  throw new NotImplementedException(); }

erride bool EOF

  throw new NotImplementedException(); }

erride string GetAttribute(int i)

  new NotImplementedException();

erride string GetAttribute(string name, string
```

Figure 5-10: IntelliSense adds the methods from the abstract class to the concrete class.

Implementing interfaces

An *interface* is a data type that lists a set of methods. The interface doesn't actually provide the code for the methods. The implementation of the interface occurs within a class you create. Many classes can implement a single interface, and any class can implement multiple interfaces. The interface creates a contract that defines what the methods look like in a class.

To implement interfaces using IntelliSense, follow these steps:

1. **Create a new class.**
2. **Type a colon (:) after the class name declaration.**
3. **Type the name of the interfaces you want to implement.**

   For example, the following class header implements the `IFormatProvider` and `ICustomFormatter` interfaces:

   ```
   public class RssReader: IFormatProvider, ICustomFormatter
   ```

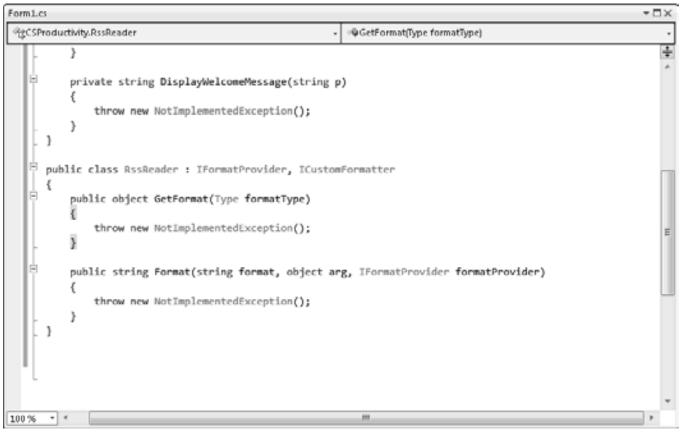
4. **Click the interface name in the header.**

   A shortcut menu appears.

5. **Choose Implement Interface from the shortcut menu.**

   IntelliSense adds stubs for the methods from the interface.

   Figure 5-11 shows a class that implements the `IFormatProvider` and `ICustomFormatter` interfaces.



```
    }

    private string DisplayWelcomeMessage(string p)
    {
        throw new NotImplementedException();
    }
}

public class RssReader : IFormatProvider, ICustomFormatter
{
    public object GetFormat(Type formatType)
    {
        throw new NotImplementedException();
    }

    public string Format(string format, object arg, IFormatProvider formatProvider)
    {
        throw new NotImplementedException();
    }
}
```

Figure 5-11: IntelliSense stubs out methods when you implement an interface.

Overriding members

At times, you want to provide a different implementation of a method in your new class than what's provided in the base class. For example, say that you create a new class `Dog` that inherits from the base class `Animal`. `Animal` class provides a `Vocalize` method that defines what it means for an animal to "speak." You want to change the implementation of `Vocalize` in your `Dog` class so that you can make the dog bark.

Providing another implementation of a method requires two elements:

- **In the base class,** the method must use the keyword `virtual`. For example, the following code creates a `virtual` method in the `Animal` class:

  ```
  public virtual void Vocalize()
  {
    //some code here
  }
  ```

- **In the derived class,** the method must use the keyword `override`. The following code creates a new implementation of the base class's `virtual` method created in the derived class `Dog`:

  ```
  public override void Vocalize()
  {
    base.Vocalize();
  }
  ```

You can use IntelliSense to display a list of methods that can be overridden and create method stubs for you. To use IntelliSense to override methods, follow these steps:

1. **Create a new class.**

   All classes implicitly derive from `System.Object`. As a result, your class can override methods of `System.Object`. See Chapter 3 in Book III for recommendations on overriding `System.Object`.

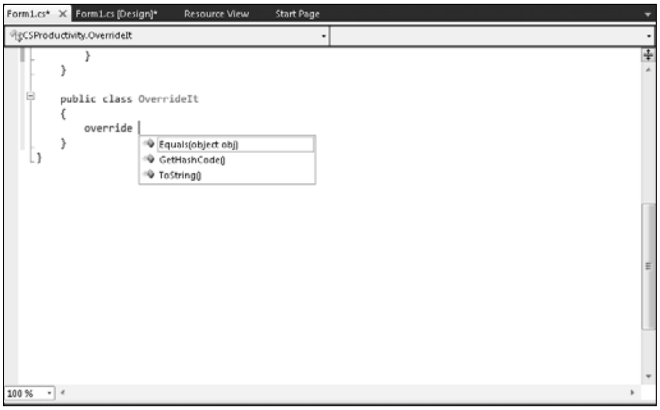
2. **Position your cursor in the Code Editor where you want to insert the new method.**

3. **Type the keyword** override **and then press the spacebar.**

   A list of methods that you can override appears, as shown in Figure 5-12.

4. **Select a method to override.**

   The method stub appears in the Code Editor.



```
        }
    }

    public class OverrideIt
    {
        override |
    }
}
```

```
Equals(object obj)
GetHashCode()
ToString()
```

Figure 5-12: Use IntelliSense to select a virtual method to override.

Creating XML Documentation

You probably know that you're supposed to comment and document your code. If you're like most developers, you probably don't take the time to actually do it. IntelliSense and the .NET language compilers make it easy for you to turn your lowly code comments into documentation.

Creating documentation for your code requires you to use the following:

- **Comment markers:** Each language has its own comment marker for creating documentation. The comment marker in Visual Basic is three, single-quotation marks (`'''`). The C# comment marker is three forward slashes (`///`).

- **XML tags:** Each language defines a set of recommended XML tags that you should use to define your comments, but you can use any well-formed XML tag you want. Table 5-2 lists some of the recommended XML tags for Visual Basic and C#.

Table 5-2 Code Documentation XML Tags

XML Tag	*Description*
`<c>`	Marks comment as code
`<code>`	Marks multiple lines of comments as code
`<example>`	Identifies comment as an example of how to use the documented code
`<exception>`	Shows exceptions that can be thrown
`<include>`	Refers to another file containing documentation
`<list>`	Creates a bulleted or numbered list or table in your documentation
`<para>`	Denotes a paragraph
`<param>`	Describes a parameter used in a method declaration
`<paramref>`	Creates a reference in your documentation to a parameter
`<permission>`	Grants document access permissions to a member
`<remarks>`	Adds supplementation information
`<returns>`	Describes a return value
`<see>`	Creates a hyperlink to another type or member
`<seealso>`	Specifies text to appear in a See Also section
`<summary>`	Provides a brief description of the type or member
`<typeparam>`	Describes a type parameter for a generic type
`<typeparamref>`	Refers to a type parameter for a generic type
`<value>`	Describes the value a property represents

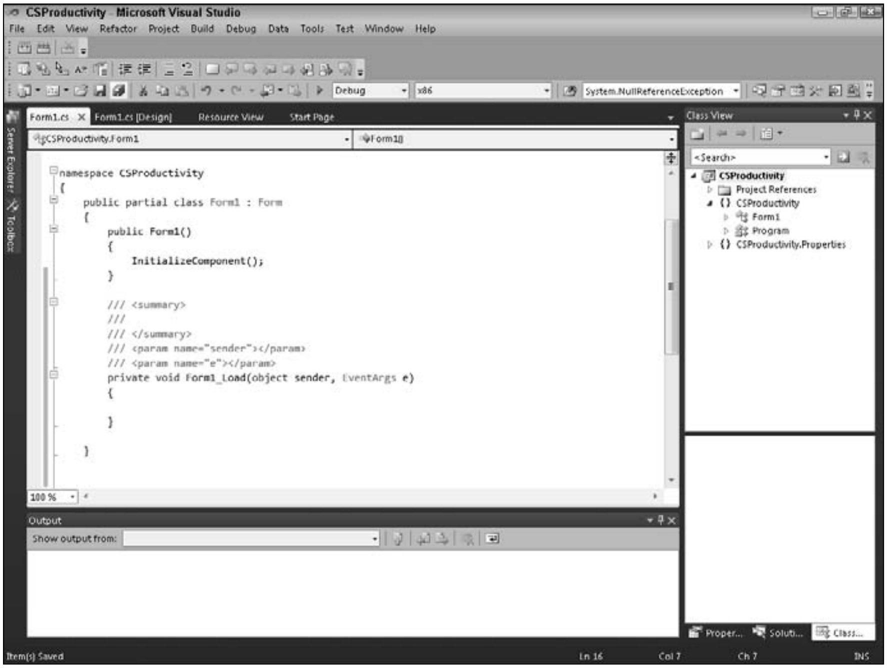
Like all XML tags, you place your text inside opening and closing tags. For example, the code that follows shows a code comment in Visual Basic and the class header it describes:

```
''' <summary>
'' This class provides a Hello World greeting.
''' </summary>
''' <remarks>Created as an example.</remarks>
Public Class HelloWorld
```

The Code Editor provides a documentation template when you add comment markers. To add documentation comments to your source code, do the following:

1. **Position your cursor above the type or member you wish to document.**
2. **Type the comment marker for your programming language.**

   A documentation template appears in your source code, as shown in Figure 5-13.
3. **Type your comments in between the opening and closing XML tags.**



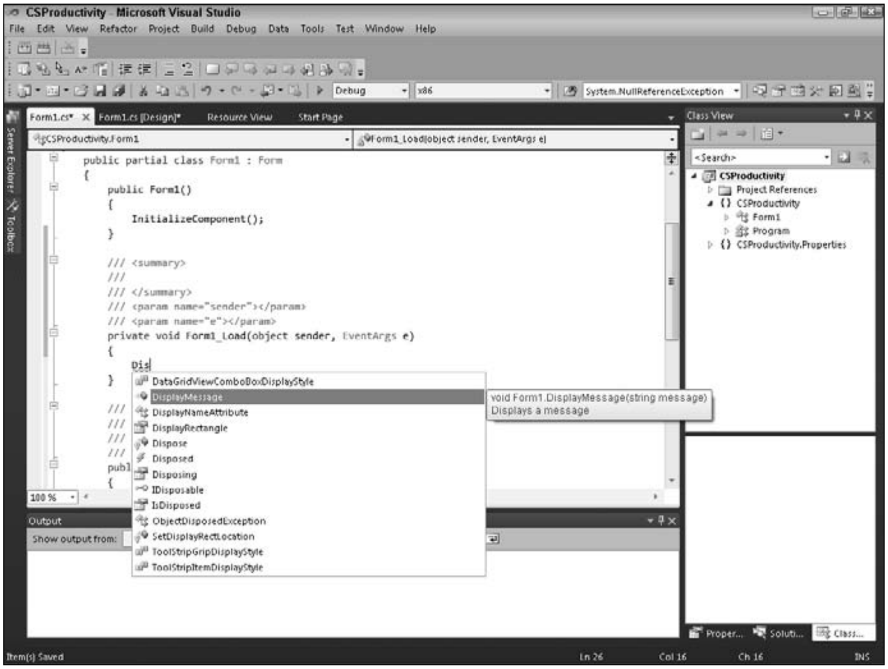
```
namespace CSProductivity
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void Form1_Load(object sender, EventArgs e)
        {

        }

    }
```

Figure 5-13: The Code Editor adds a document template when you type code comment markers.

The documentation template provided depends on whether you're documenting a type or a member. For example, Visual Basic supplies the following template for a method:

```
''' <summary>
'''
''' </summary>
''' <returns></returns>
''' <remarks></remarks>
```

The code comments you add appear in IntelliSense, or you can output the comments as XML documentation. Figure 5-14 shows an example of a code comment in the IntelliSense List Members feature.



```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    /// <summary>
    ///
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void Form1_Load(object sender, EventArgs e)
    {
        Dis
    }

    ///
    ///
    ///
    ///
    publ
    {
```

void Form1.DisplayMessage(string message)
Displays a message

Figure 5-14: Code comments appear in the IntelliSense List Members feature.

Outputting your code comments as XML documentation is a feature of your language's compiler. You can turn on XML documentation in the Visual Studio Project Designer. To enable XML documentation using Project Designer, do the following:

1. **Right-click your project in Solution Explorer.**

   A shortcut menu appears.

2. **Choose Properties.**

   Project Designer appears.

3. **Click the Compile tab in a Visual Basic project. In a C# project, click the Build tab.**

4. **In Visual Basic, enable the Generate XML Documentation File check box. In C#, enable the XML Documentation File and the Specify a Filename check boxes.**

   By default, the XML documentation file is saved to the bin folder for the build configuration selected.

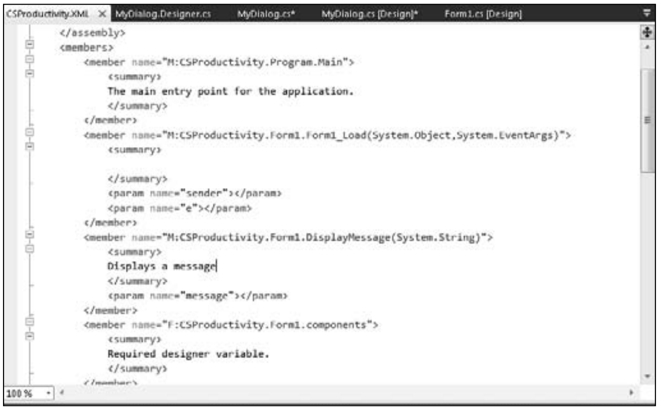
5. **Click the X in the upper-right corner of Project Designer to close the window.**

The preceding set of steps tells the compiler to use the `/doc` switch. You can add the `/doc` switch to any batch files or build scripts you use to automatically generate output.

The preceding set of steps doesn't work for ASP.NET Web sites because Web sites don't use the project model. You have a few options, which include

- **Precompile your site.** ASP.NET Web sites are compiled as resources on the site and accessed by users. You can, however, precompile the site, which would allow you to the `/doc` option on the compiler.
- **Use Web Application Projects.** Microsoft has released an extension to Visual Studio 2008 that allows you to use projects with your ASP.NET Web sites.

To create XML documentation for your project, build your project. Any code decorated with comment markers is output to an XML file. Figure 5-15 shows a sample of XML documentation.



```
    </assembly>
    <members>
        <member name="M:CSProductivity.Program.Main">
            <summary>
            The main entry point for the application.
            </summary>
        </member>
        <member name="M:CSProductivity.Form1.Form1_Load(System.Object,System.EventArgs)">
            <summary>

            </summary>
            <param name="sender"></param>
            <param name="e"></param>
        </member>
        <member name="M:CSProductivity.Form1.DisplayMessage(System.String)">
            <summary>
            Displays a message
            </summary>
            <param name="message"></param>
        </member>
        <member name="F:CSProductivity.Form1.components">
            <summary>
            Required designer variable.
            </summary>
        </member>
```

Figure 5-15: The compiler creates an XML documentation file based on your code comments.

Because the documentation files use XML output, you can use Extensible Stylesheet Language Transformation (XSLT) style sheets to transform your XML into any format you want, including HTML or plain text.

You can create MSDN-style documentation from the XML documentation files created by your language's compiler. NDoc is a tool that creates documentation for C# projects. You can download NDoc for free at `http://ndoc.sourceforge.net`

VBCommenter creates documents for Visual Basic projects. Download VBCommenter from MSDN at

`http://code.msdn.microsoft.com/VBCommenter`

Read the associated article at

`http://msdn2.microsoft.com/en-us/library/aa289191(VS.71).aspx`

Factoring in the Refactoring

Refactoring is changing source code from its current form to another. Even if you think you've never refactored code, chances are that you have. You refactor any time you make a decision about how to implement something. Say that you need to write a program that processes data from a file. You start jotting down ideas for procedures and variables you know you need. As you commit your ideas to code, you invariably break your code into smaller, reusable bits of code. That's refactoring.

Refactoring is often necessary when you must add a feature or fix a bug in an existing code base. You refactor code any time you ask yourself whether you should

- Copy and paste code or extract common code to a reusable procedure.
- Use a lengthy conditional statement or create separate procedures.
- Write a single procedure that performs three tasks or write one procedure that calls three other procedures.

Refactoring is quite common in object-oriented programming. Before OOP, programs were often *brittle.* That is, it was difficult for a developer to make a change to an existing code base without fear of triggering a cascade of bugs throughout the rest of the program. OOP provides many constructs that make it easier to break units of code into smaller, reusable pieces that are loosely coupled to one another, which makes refactoring possible.

Of course, this describes the goal of OOP techniques. In reality, most programmers don't start out implementing perfect, object-oriented designs using OOP. Nor should you strive to do so. The beauty of refactoring using OOP is that it gives you permission to let go of perfection. Letting go of your visions of building the perfect system allows you to get busy building the best system you can in the here and now. You can always refactor your code later when you have more knowledge.

In the past, refactoring often meant using a combination of Copy, Paste, and Find and Replace to rename parameters or change return values. Visual Studio 2010 provides support for many kinds of refactorings. Table 5-3 provides a list of common refactoring techniques and where you can find them.

Table 5-3 Refactoring Options in Visual Studio 2010

Technique	*Description*	*Tool*	*Language*
Encapsulate field	Creates a property from a public field	Refactor menu	C#
Extract interface	Creates a new interface from an existing method	Refactor menu, Class Designer	C#
Extract method	Creates a new method from a code fragment	Refactor menu	C#
Implement an abstract	Creates a derived class from an abstract class	Class Designer, IntelliSense	C#, VB
Implement an interface	Creates method stubs in a class that inherits from an interface	Class Designer, IntelliSense	C#, VB
Move a type member	Moves a type member from one type to another type	Class Designer	C#, VB
Override	Creates a method stub for a member being overridden	Class Designer, IntelliSense	C#, VB
Promote local variable	Changes a local variable to a method parameter	Refactor menu	C#
Remove parameters	Removes parameters from a method's declaration	Refactor menu	C#
Rename identifiers	Renames a type or a member of a type	Class Designer	C#, VB
Reorder parameters	Changes the order of a method's parameters	Class Designer, Refactor menu	C#

Visual Studio 2010 updates all the places in source code where the refactored code is called. There's no need to use Find and Replace to update method calls.

The refactoring techniques implement abstract classes, implement an interface, and override features of IntelliSense. See the earlier section "It's Not Your Father's IntelliSense Any More" for examples of how to use it.

C# provides more refactoring opportunities than Visual Basic. Your choices for Visual Basic are mostly limited to those features available in IntelliSense and Class Designer.

Refactoring with Class Designer

Class Designer provides access to the IntelliSense refactoring features via the Class Designer menu. The Class Designer menu adds a Refactoring menu that you can use to access refactoring features specific to C#.

You aren't restricted to using the menu items to refactor in Class Designer. For example, to implement an interface, follow these steps:

1. **Add a class that you want to implement the interface.**
2. **Add the interface you want to implement.**
3. **Click the Inheritance icon in the toolbox.**
4. **Drag the line from the class to the interface.**

   The interface's members are stubbed out in the class, as Figure 5-16 shows.

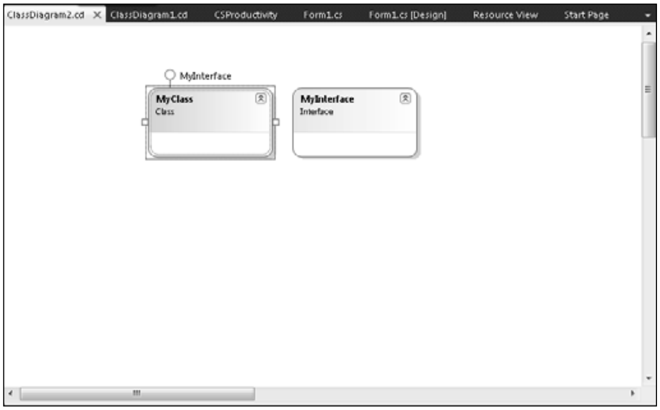


Figure 5-16: Drag an inheritance line to implement an interface in Class Designer.

Moving a type member to another type in Class Designer is as simple as cut and paste. For example, say that you create a `Customer` class. While you're adding the properties `FirstName`, `LastName`, and `CompanyName`, you realize that none of your customers have a `CompanyName` and a `FirstName` and LastName. You decide to create two new classes that inherit from the Customer class. You want to move the FirstName and LastName properties to a ResidentialCustomer class and the CompanyName property to a CommercialCustomer class. To move the types, you do the following:

1. **Right-click the property in the class you want to move and, from the shortcut menu that appears, choose Cut.**

   The property disappears from the class.

2. **Right-click the class where you want to move the property and, from the shortcut menu that appears, choose Paste.**

   The property appears in the class.

Refactoring C#

Visual Studio provides extensive support for refactoring in the C# programming language. You can access all the C# refactoring commands by clicking the Refactor menu while in the Code Editor. A limited set of commands is available in Class Designer.

In most cases, the C# refactoring commands work like this:

1. **Position your cursor on the source code that you wish to refactor.**

   On commands that change methods, such as Remove Parameters, you can place your cursor on the method declaration or the method call.

2. **Choose the refactoring command you want to execute from the Refactor menu.**

   A dialog box appears.

3. **Complete the dialog box for the refactoring command.**

Some commands, such as the Reorder Parameters command, display a preview dialog box where you can review changes before you commit.

For example, to reorder a method's parameters:

1. **Position your cursor on the method declaration or the method call.**
2. **Choose Refactor➪Reorder Parameters.**

   The Reorder Parameters dialog box appears.

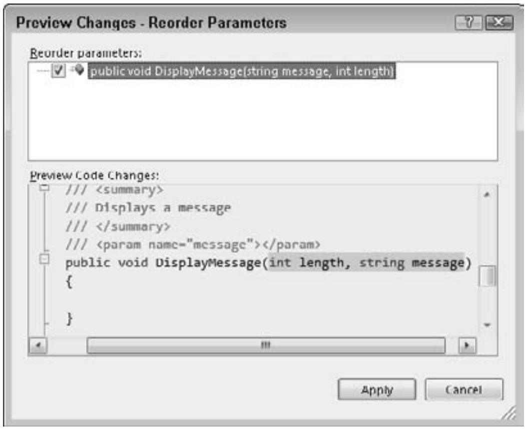
3. **Click the arrows to reorder the parameters.**

   A preview of the new method signature appears in the dialog box.

4. **Enable the Preview Reference Changes check box.**
5. **Click OK.**

   The Preview Changes window appears, as shown in Figure 5-17.

   The Preview Changes window displays the method signature and each method call. The changed code appears in the lower pane.
6. **Click the method's references to review code changes.**
7. **Click Apply to save the refactored code.**



☑ public void DisplayMessage(string message, int length)

```
/// <summary>
/// Displays a message
/// </summary>
/// <param name="message"></param>
public void DisplayMessage(int length, string message)
{

}
```

Figure 5-17: Preview your refactored code in the Preview Changes window.

Chapter 6: Exploring Web Services

In This Chapter

- **Creating a new Web service**
- **Testing Web services with `GET`, `POST`, and `SOAP`**
- **Using WSDL and UDDI to find Web services**
- **Consuming Web services in server-side code**
- **Accessing Web services with client-side code**

Never created a Web service before? Never fear. This chapter shows you how to use Visual Studio 2010 to create a Web service, test it, and call it from another application.

Defining a Web Service

A *Web service* is a kind of Web application. Unlike regular Web applications, however, a Web service has no user interface. Generally, Web services aren't intended to be accessed by end users directly. Rather, Web services are consumed by other applications. For example, a Web service that returns the current temperature for a given city could be called by — or consumed by — a town's Web page.

Web services have two basic uses:

- **Interface between systems:** Many different kinds of systems run on different hardware and platforms. Trying to write software that allows all these disparate systems to talk to each other has been challenging. Many of the interfaces quickly become brittle: That is, any small change in the interface makes communication difficult, unreliable, or impossible. Web services use standards that overcome the difficulties of creating system interfaces.
- **Reusable components:** Rather than copy and paste code or deal with distributing components, you can make the features of your code available as a Web service.

Don’t be fooled by the term *Web service.* It’s so called because it uses the technologies of the Web: namely, HyperText Transfer Protocol (HTTP) and eXtensible Markup Language (XML). However, you aren’t restricted to publishing your Web services over the Internet. Many companies use Web services within their companies. For example, your public relations department can publish its press releases via a Web service method called `GetPressReleases(date)`. Your Web site development staff can consume those services and display them on a public Web site, a corporate portal, or a really simple syndication (RSS) feed.

Saying Hello to Web Services

Web services make extensive use of Web-based technologies, such as HTTP and XML. Thankfully, you don’t need to dig into how Web services use these technologies because Visual Studio and ASP.NET take care of all that for you.

To create a new Web service, follow these steps:

1. **Choose File⇨New⇨Web Site.**

   The New Web Site window appears.

2. **Click the ASP.NET Web Service template.**
3. **Type a name for your Web service.**

   Give your Web service a meaningful name.

4. **Click OK.**

   Visual Studio creates the Web service project.

The Web service created by Visual Studio is fully functional. To run the Web service, press Ctrl+F5. Visual Studio launches the service in your Web browser.

Web services don’t have user interfaces. ASP.NET uses a template to generate the `service.asmx` page. The page lists a single method, `HelloWorld`, created by Visual Studio.

To test the `HelloWorld` method, do the following:

1. **Click the HelloWorld hyperlink on the `service.asmx` page.**

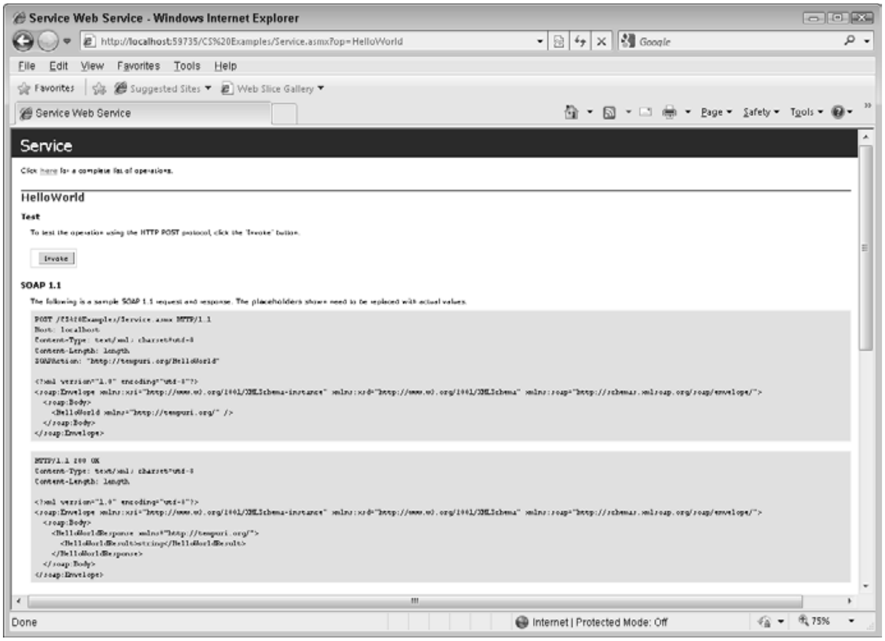
   A test page appears, as shown in Figure 6-1.

   The page includes a button to invoke the test method. It also shows the sample Simple Object Access Protocol (SOAP) headers that ASP.NET uses to communicate with the Web service using SOAP. You don’t need to know SOAP to perform basic tasks with Web services because Visual Studio and ASP.NET take care of all that for you. However, a knowledge of SOAP is very helpful when you want to perform advanced tasks and need to debug your application. SOAP is explained in more detail in the upcoming section "Testing Web services."

2. **Click the Invoke button.**

   An XML file appears with the value `Hello World` in a single string node.

To stop testing the Web service, right-click the ASP.NET Development Server icon in the Notification and choose Stop from the contextual menu. The Web service becomes unavailable at this point.


Service Web Service - Windows Internet Explorer
http://localhost:59735/CS%20Examples/Service.asmx?op=HelloWorld
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Service Web Service
Page
Safety
Tools
Done
Internet | Protected Mode: Off
75%

Service

Click here for a complete list of operations.

HelloWorld

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Invoke

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /CS%20Examples/Service.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/HelloWorld"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <HelloWorld xmlns="http://tempuri.org/" />
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <HelloWorldResponse xmlns="http://tempuri.org/">
      <HelloWorldResult>string</HelloWorldResult>
    </HelloWorldResponse>
  </soap:Body>
</soap:Envelope>
```

Figure 6-1: ASP.NET generates a page to test the Web service's methods.

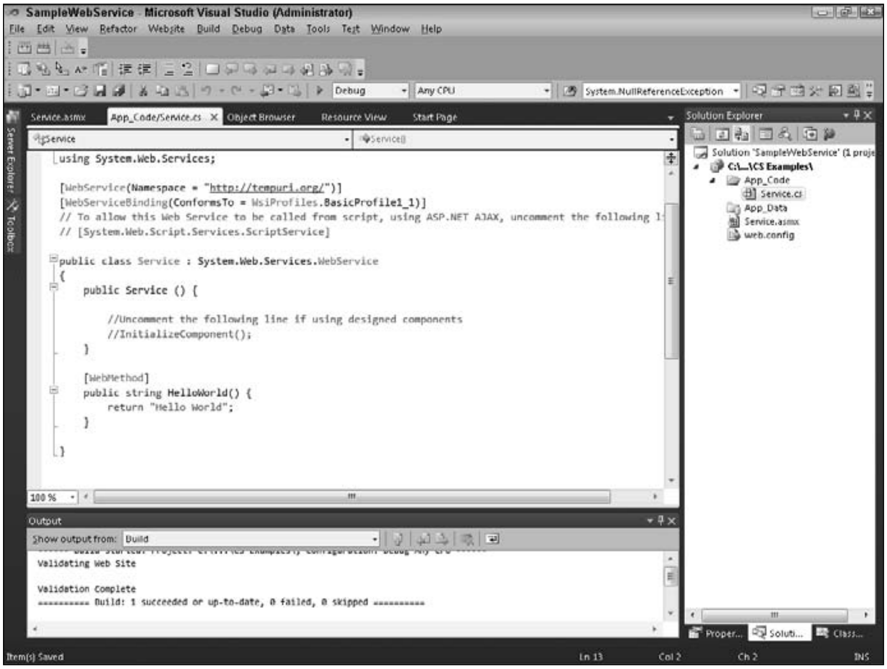
Understanding ASP.NET Web services

A Web service project comprises two main components: an entry point to access the service and a class that contains the code for the Web service. Because Web services don't have user interfaces like Web sites, Web services in ASP.NET are accessed via an `.asmx` page, which serves as the entry point to the Web service. An `.asmx` page has a `WebService` processing directive at the top of the page that defines the programming language of the Web service, the code-behind file, and the class name that provides the Web service's functionality. The features of a Web service are implemented in a class. The class uses attributes to identify which methods are publicly accessible via the Web service. Using attributes allows you to use private methods in your class.

TIP

Although most Web services aren't accessed via `.asmx` pages, ASP.NET Web sites are typically accessed via `.aspx` pages.

Figure 6-2 shows an example of a simple `HelloWorld` Web service created with Visual Studio.



```
using System.Web.Services;

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following l
// [System.Web.Script.Services.ScriptService]

public class Service : System.Web.Services.WebService
{
    public Service () {

        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

    [WebMethod]
    public string HelloWorld() {
        return "Hello World";
    }

}
```

```
Validating Web Site

Validation Complete
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Figure 6-2: A Web service includes an entry point and a class.

Visual Studio automatically wires up the Web service's class with the attributes that it needs to function as a Web service. The class that Visual Studio creates for the Web service uses two attributes:

- `WebService`: A Web service has one `WebService` attribute that identifies the Web service's namespace. The attribute can also include a description.

✦ `WebMethod`: Each method that's accessible from the Web service must have the `WebMethod` attribute above it.

To change the name of the service and the class created by Visual Studio, follow these steps:

1. **Right-click the `.asmx` file in Solution Explorer.**
2. **Choose Rename from the shortcut menu.**
3. **Type a new name for the `.asmx` file, such as HelloWebService.asmx.**

   The name you use for the `.asmx` file is the name you use to access the Web service via its URL.
4. **Repeat Steps 1 through 3 to rename the `Service.vb` or `Service.cs` class created by Visual Studio.**

   The class is located in the `App_Code` folder.

   The file extension depends on whether you're using Visual Basic (`.vb`) or C# (`.cs`) as the programming language.
5. **Open the class file and change the class' name from Service to its new name, such as `HelloWebService`.**
6. **Open the `.asmx` file and change the `WebService` processing directive so that it points to the new class file and name, as shown in the following code:**

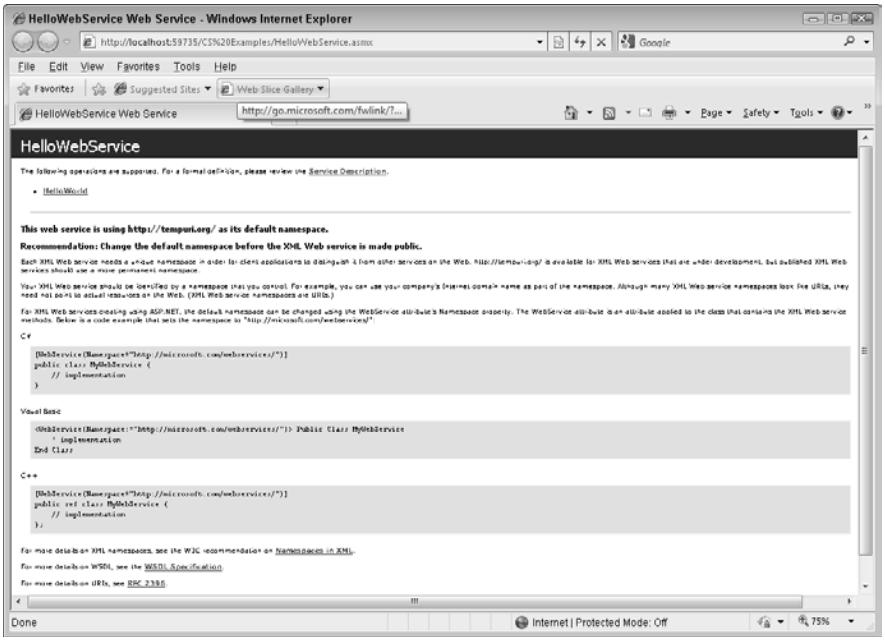
   ```
   <%@ WebService Language="C#" CodeBehind="~/App_Code/HelloWebService.cs"
       Class="HelloWebService" %>
   ```

   An `.asmx` file can point to more than just a class file. It can also point to a precompiled assembly, or you can include your code inside the `.asmx` file.
7. **Press Ctrl+F5 to run the Web service.**

   Notice that the Web service's name and URL are updated, as shown in Figure 6-3.

Each Web service must have a unique namespace. The namespace qualifies the Web service's methods so that each method is unique. Visual Studio automatically assigns the namespace `http://tempuri.org/` to Web services. You should change the namespace. The namespace doesn't have to point to an actual working URL. Rather, it should be unique. However, some Web services do use a URL that points to a Web site with useful information about the Web service.

The default namespace `http://tempuri.org/` is pronounced *TEMP-you-are-eye,* which is short for *temporary URI.* A URI — Uniform Resource Identifier — is used to provide a name or location for a resource.


HelloWebService Web Service - Windows Internet Explorer
http://localhost:59735/CS%20Examples/HelloWebService.asmx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
HelloWebService Web Service
http://go.microsoft.com/fwlink/?...
Page Safety Tools
Done
Internet | Protected Mode: Off
75%

HelloWebService

The following operations are supported. For a formal definition, please review the Service Description.

- HelloWorld

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":

C#

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // implementation
}
```

Visual Basic

```
<WebService(Namespace:="http://microsoft.com/webservices/")> Public Class MyWebService
    ' implementation
End Class
```

C++

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public ref class MyWebService {
    // implementation
};
```

For more details on XML namespaces, see the W3C recommendation on Namespaces in XML.

For more details on WSDL, see the WSDL Specification.

For more details on URIs, see RFC 2396.

Figure 6-3: Change the name of the Web service to make it more descriptive.

To change the namespace and add a description to the Web service, follow these steps:

1. **Open the class file used by your service.**

   The class file is in the `App_Code` folder.

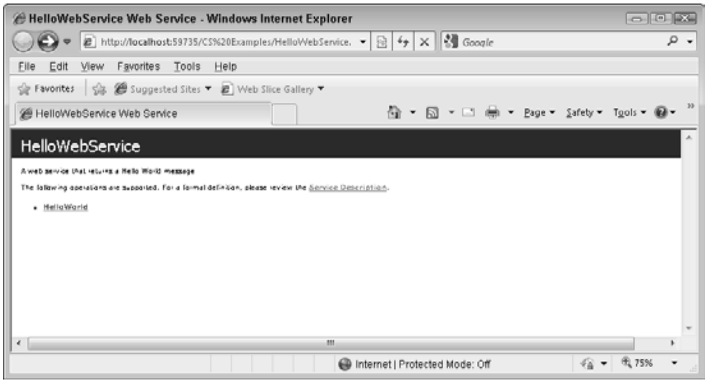
2. **Type a new namespace in place of the default namespace and add a description for the Web service, as shown in the following:**

   ```
   [WebService(Namespace="http://mycompany.com/webservices", Description= "A
       web service that returns a Hello World message")]
   ```

   The `WebMethod` attribute supports the `Description` property. Use it to add a description to the methods exposed by your Web service.

3. **Press Ctrl+F5 to run the Web service.**

   The Web service's description appears on the ASP.NET-generated page, as shown in Figure 6-4. Notice also that a description for the `HelloWorld` method appears.


HelloWebService Web Service - Windows Internet Explorer
http://localhost:59735/CS%20Examples/HelloWebService.
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
HelloWebService Web Service
Page
Safety
Tools
Internet | Protected Mode: Off
75%

HelloWebService

A web service that returns a Hello World message

The following operations are supported. For a formal definition, please review the Service Description.

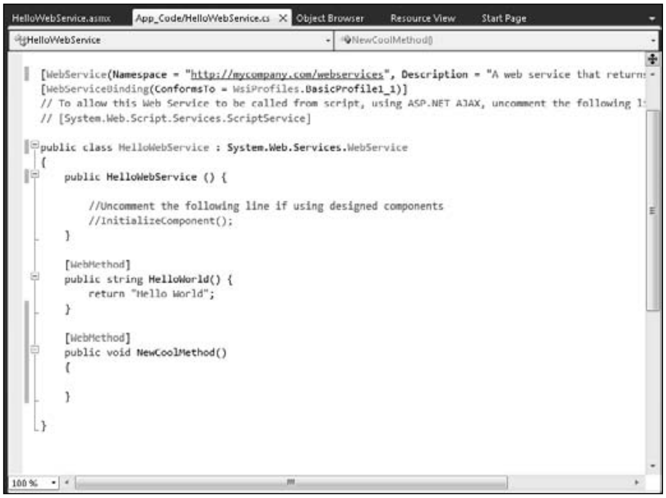
- HelloWorld

Figure 6-4: Use the `Description` property to add descriptions to your Web service and methods.

Adding a method

The methods that you add to your Web service class aren't automatically accessible via your Web service. You must explicitly mark them as Web service methods.

You mark Web service methods with a `WebMethod` attribute. Figure 6-5 shows a code sample with a single `public` method that uses two `private` methods. The `public` method is marked with the `WebMethod` attribute. Note that the `private` methods aren't accessible.



```
[WebService(Namespace = "http://mycompany.com/webservices", Description = "A web service that return:
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following l:
// [System.Web.Script.Services.ScriptService]

public class HelloWebService : System.Web.Services.WebService
{
    public HelloWebService () {

        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

    [WebMethod]
    public string HelloWorld() {
        return "Hello World";
    }

    [WebMethod]
    public void NewCoolMethod()
    {

    }

}
```

Figure 6-5: Use the `Web Method` attribute to make a `public` method accessible via a Web service.

Testing Web services

Short of building a full-blown application, you have several ways to test your Web services. The easiest way to test your Web services is to press F5 while in Visual Studio. You can also access your Web services via its URL or by using an HTTP form. In most cases, you communicate with Web services via HTTP (the communication protocol of the Internet).

Whichever method you choose, you must be aware of your three communication choices when working with Web services:

- `GET`**:** Uses HTTP with an encoded URL to call the Web service. This method is very much like the Representative State Transfer (REST) technique used by many public Web services — it has the advantage of being incredibly easy to use and test. The basic syntax for accessing a Web service with `GET` is to use a URL like the following:

  ```
  http://<servername>/<projectname>/service.asmx/
     methodname?parameter1=value&parameter2=value
  ```

- `POST`**:** Uses HTTP to pass the method's parameters without displaying them in the URL. The `POST` approach is more secure.
- `SOAP`**:** Uses an XML dialect called SOAP to enclose the request to and response from the Web service. `SOAP` is often used in conjunction with HTTP, although it isn't required. `SOAP` is the standard for Web services message encapsulation.

The testing services provided by ASP.NET use `POST` by default. The following section shows you how to enable testing with `GET`. ASP.NET doesn't provide a facility for testing `SOAP` messages. You must create your own. To read more about using SOAP, see the section "Communicating with a Web service," later in this chapter.

Sending a GET request

To see a GET request at work, you must modify the default code provided with the Web service. The following code shows the modified version:

```
<WebMethod()> _
Public Function HelloWorld(ByVal name As String) _
   As String

    ' Supply a default value if the value is blank.
    If name.Length = 0 Then
        name = "World"
    End If

    ' Return a hello string.
    Return "Hello " & name
End Function
```

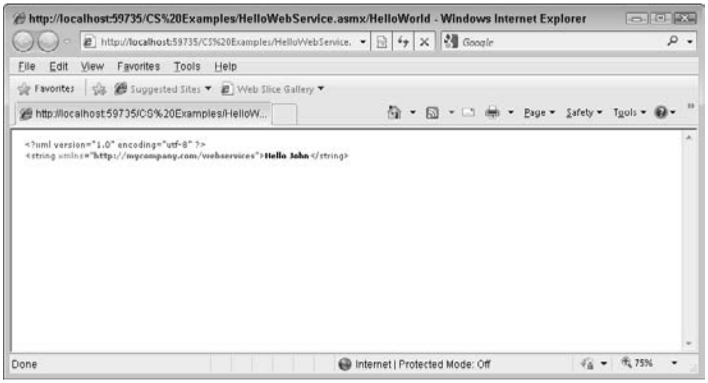
Visual Studio doesn't initially provide HTTP GET, so you need to enable the GET protocol in your project's web.config file. To add the protocol, paste the following code between the opening and closing <system.web> tags in web.config:

```
<webServices>
   <protocols>
      <add name="HttpGet"/>
   </protocols>
</webServices>
```

A GET request passes the Web services parameters with the URL. For example, the following URL calls the HelloWorld method on the HelloWebService and passes the value John to the name parameter:

```
http://localhost:49166/WebService/HelloWebService.asmx/
   HelloWorld?name=John
```

The port that your ASP.NET Development Server uses will vary from these because Visual Studio generates a random number for each project. To see the port number for your setup, hover the mouse cursor over the ASP.NET Development Server icon in the Notification area and use that port number in the GET URL you create. The Web service returns an XML document with the value Hello John, as shown in Figure 6-6.


http://localhost:59735/CS%20Examples/HelloWebService.asmx/HelloWorld - Windows Internet Explorer
http://localhost:59735/CS%20Examples/HelloWebService.
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
http://localhost:59735/CS%20Examples/HelloW...
Page
Safety
Tools
Done
Internet | Protected Mode: Off
75%

```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://mycompany.com/webservices">Hello John</string>
```

Figure 6-6: The Web service returns an XML document.

Passing in parameters at the end of a URL is *encoding.* Any time you see a URL with parameters appended, your browser sends an HTTP GET request to the Web server. GET requests are considered more risky than POST requests because the parameter values are in plain sight. The alternative is to use an HTTP POST request, which sends the parameter values in the message body.

Sending a POST request

You can use an HTML form to test your Web service by using a POST request. With a POST request, the Web service's parameters aren't visible in the URL.

To test a Web service with an HTTP form, follow these steps:

1. **Click the Web service in Solution Explorer.**

   Press Ctrl+Alt+L to open Solution Explorer if it's closed.

2. **Choose Website➪Add New Item.**

   The Add New Item window appears.

3. **Click the HTML Page icon.**

4. **Type a name for the HTML page and then click Add.**

   The HTML page appears in the Code Editor.

5. **Type the following HTML to create a form:**

   ```
   <form method="POST" action='http://<servername>/
      <projectname>/HelloWebService.asmx/HelloWorld'>
   </form>
   ```

   The action attribute specifies where the form should post. Type the URL of your Web service. Append a forward slash and the name of the method to execute. In the preceding example, the form executes the HelloWorld method on the Web service.

6. **Drag and drop input elements for text from the toolbox onto the HTML page.**

7. **Drag and drop a Submit button (note that the button type is Submit, even though the caption reads button).**

   Make sure that the elements appear between the form tags.

8. **Set the name attribute for your input elements to the parameter for your method.**

   For example, the following HTML form posts to the HelloWorld method of the HelloWebService (note that the action attribute appears on a single line in your code):

   ```
   <form method="POST"
      action='http://localhost:49166/WebService/
         HelloWebService.asmx/HelloWorld'>

        Enter Your Name:
        <input id="Text1" type="text" name="name"/>
        <input id="Button1" type="submit" value="button" />
   </form>
   ```

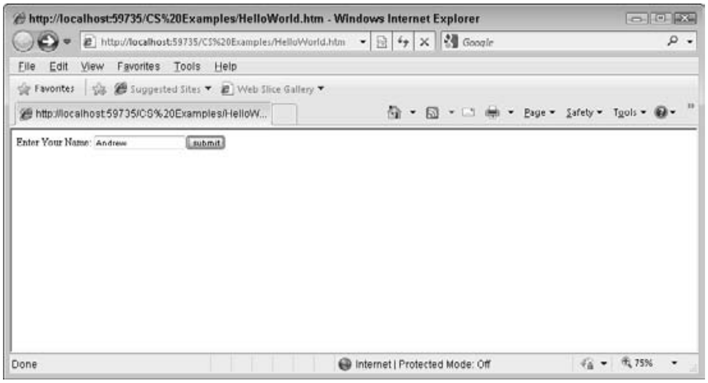
9. **Press Ctrl+F5 to run your Web page.**

   The page appears.

   Make sure that you have the HTML page open when you press Ctrl+F5 or set the HTML page as the startup page for the site. Otherwise, you'll see the Web service test page open, rather than the HTML page. If you make a mistake, simply select the HTML page and press Ctrl+F5.

10. **Type the parameter values in the page, as shown in Figure 6-7.**

11. **Click button.**


http://localhost:59735/CS%20Examples/HelloWorld.htm - Windows Internet Explorer
http://localhost:59735/CS%20Examples/HelloWorld.htm
Google
File
Edit
View
Favorites
Tools
Help
Favorites
Suggested Sites
Web Slice Gallery
http://localhost:59735/CS%20Examples/HelloW...
Page
Safety
Tools
Done
Internet | Protected Mode: Off
75%

Enter Your Name: Andrew submit

Figure 6-7: Type the parameter values in to the test page.

When you click the Submit button, the HTML page sends the parameters in the form `parameter=value` to the Web service method. Using the example in Figure 6-7, the browser submits `name=John` to the Web service.

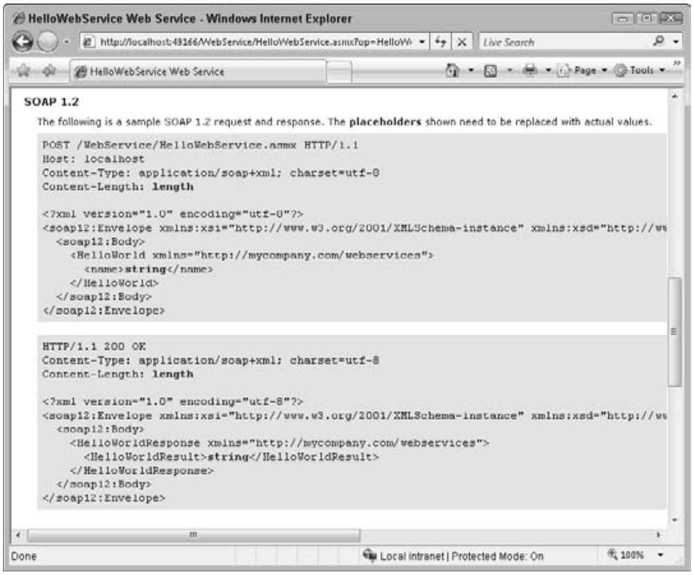
The Web service responds with an XML document containing the value Hello John, as shown in Figure 6-6.

See the upcoming section "Consuming a Web Service" for more information on what to do with the XML document returned by the Web service.

Testing with SOAP

You might also want to test the SOAP messages generated by your Web service. The test page generated for your Web service shows a sample SOAP message that ASP.NET generates, as shown in Figure 6-8.

Because SOAP messages are XML-based, you can use standard Web technologies to test the messages. You'll find an excellent SOAP tutorial at `www.w3schools.com/soap/default.asp`. The following example uses JScript and Microsoft eXtensible Markup Language (MSXML) Core Services to submit a SOAP request and display the response. *MSXML* is the Microsoft XML parser. You use it to process the XML you send and receive. *JScript* is a client-side scripting language that you use to display values on the Web page. The following example shows you how to use MSXML and JScript to test SOAP messages without assuming that you have any knowledge of these technologies.


HelloWebService Web Service - Windows Internet Explorer
http://localhost:49166/WebService/HelloWebService.asmx?op=HelloW
Live Search
HelloWebService Web Service
Page
Tools
Done
Local intranet | Protected Mode: On
100%

SOAP 1.2

The following is a sample SOAP 1.2 request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /WebService/HelloWebService.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://ww
  <soap12:Body>
    <HelloWorld xmlns="http://mycompany.com/webservices">
      <name>string</name>
    </HelloWorld>
  </soap12:Body>
</soap12:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://ww
  <soap12:Body>
    <HelloWorldResponse xmlns="http://mycompany.com/webservices">
      <HelloWorldResult>string</HelloWorldResult>
    </HelloWorldResponse>
  </soap12:Body>
</soap12:Envelope>
```

Figure 6-8: ASP.NET generates sample SOAP messages.

Here's how to create a test for SOAP:

1. **Add a new HTML page to your project.**
2. **Add a `script` tag to the page between the `head` tags, as shown in the following:**

   ```
   <head>
   <script type="text\jscript">
   </script>
   </head>
   ```

3. **Create a new object to hold an XML document:**

   ```
   var xmlDoc = new ActiveXObject("Msxml2.DOMDocument");
   ```

4. **Create a new object to communicate with a Web server:**

   ```
   var xmlHTTP = new ActiveXObject("Msxml2.XMLHTTP");
   ```

5. **Create a new function that creates the SOAP message and loads it into the XML document that you create in Step 3:**

```
function createXmlDoc(name)
{
   var soapXml  = "<?xml version=\"1.0\" ?>" ;
   soapXml += "<soap12:Envelope ";
   soapXml += "xmlns:xsi=\"http://www.w3.org/2001/
   XMLSchema-instance\" " ;
   soapXml += "xmlns:xsd=\"http://www.w3.org/2001/
   XMLSchema\" " ;
   soapXml += "xmlns:soap12=\
   "http://www.w3.org/2003/05/soap-envelope\">" ;
   soapXml += "<soap12:Body>" ;
   soapXml += "<HelloWorld xmlns=\
   "http://mycompany.com/webservices\">" ;
           soapXml = soapXml + "<name>" + name.value  +
   "</name>" ;
   soapXml += "</HelloWorld>";
   soapXml += "</soap12:Body></soap12:Envelope>"

   xmlDoc.loadXML(soapXml) ;
}
```

You can find an example of the SOAP request on the Web service's test page. Be sure to pass any parameter values to the request. For example, here's the SOAP request generated by ASP.NET for the Web service shown in Figure 6-5:

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="
   http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap12="http://www.w3.org/2003/05/
   soap-envelope">
  <soap12:Body>
    <HelloWorld xmlns=
   "http://mycompany.com/webservices">
      <name>string</name>
    </HelloWorld>
  </soap12:Body>
</soap12:Envelope>
```

6. **Create a function that sends the SOAP request to the Web server.**

The following is an example of how to create a function:

```
function sendXml()
{
   xmlHTTP.Open ( "Post", "http://localhost:4251/
   HelloWebService/HelloWebService.asmx", false);
   xmlHTTP.setRequestHeader("Content-Type",
   "application/soap+xml; charset=utf-8" );
   xmlHTTP.setRequestHeader("Content-Length", xmlDoc.
   xml.length);
   xmlHTTP.Send(xmlDoc.xml);
 }
```

The sendXML function uses the XMLHTTP object to send the SOAP request to the Web service. You can find the header information for the request on the Web service's test page. Here's the SOAP request generated for the Web service shown in Figure 6-5:

```
POST /HelloWebService/HelloWebService.
   asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length
```

7. **Create a function that outputs the SOAP request and response to the Web page:**

```
function writeResponses()
{
   SoapRequest.innerText =  xmlDoc.xml;
   SoapResponse.innerText =  xmlHTTP.responseText;
}
```

8. **Create a function that calls all three functions:**

```
function getSoap(name)
{
  createXmlDoc(name);
  sendXml();
  writeResponses();
}
```

9. **Add an HTML form that wires everything up:**

```
<form>
   <p>Enter name:<input id="inputName">
    </input></p>
   <p><input type="button" id="btn"  value="Enter"
                onclick="getSoap(inputName)"></input>
   </p>
   <p>Request:</p>
   <div id="SoapRequest"></div>
   <p>Response:</p>
   <div id="SoapResponse"></div>
</form>
```

10. **Display the page in the browser.**

    Figure 6-9 shows an example.

This script is modified from the script at

www.codeproject.com/webservices/aspwebsvr.asp

You can also use the testing facilities in Visual Studio Team System (VSTS) to create a Web test to test your Web services. See the topic "How to: Create a Web Service Test," in the Visual Studio documentation, for more information.

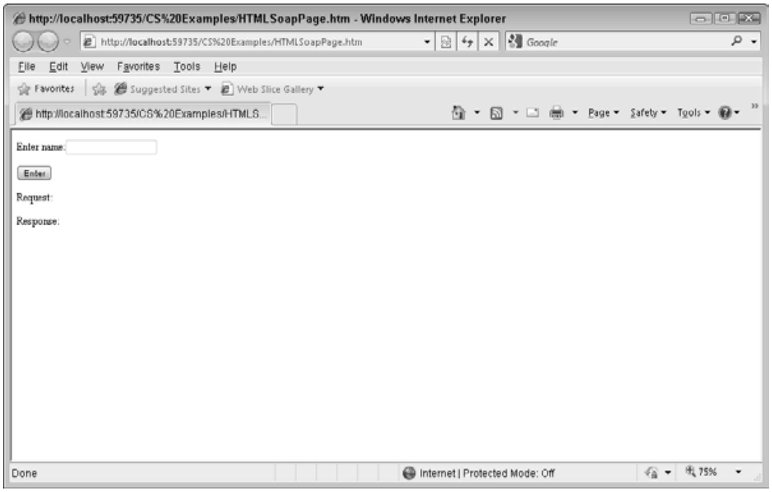

http://localhost:59735/CS%20Examples/HTMLSoapPage.htm - Windows Internet Explorer
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Page Safety Tools
Enter name:
Enter
Request:
Response:
Done
Internet | Protected Mode: Off
75%

Figure 6-9: Use a Web page to test your SOAP requests and responses.

Consuming a Web Service

Web services return XML documents. An XML document is just a string, so you can use any programming language to modify an XML document. In fact, people use all kinds of languages — from Python and PHP to JavaScript — to work with XML.

Of course, the .NET Framework provides extensive support for consuming XML and Web services, including the following:

- ✦ Visual Studio generates proxy classes that allow you to use any Web service, like a local class.
- ✦ ASP.NET generates the communication infrastructure necessary for accessing Web services.

Communicating with a Web service

Web services typically use a special message format — SOAP — that allows you to send and receive Web service requests over HTTP (the protocol of the Internet). Using SOAP is important because it's independent of both language and platform.

You aren't limited to sending `SOAP` messages over HTTP. You can use SOAP with any transport protocol.

SOAP encapsulates the messages that you send to and get from your Web service in a SOAP envelope. The SOAP request and response travels via a transport protocol, such as HTTP. As long as you have access to HTTP, you can use Web services. Think of SOAP as a delivery truck and HTTP as the highway. The contents of the SOAP delivery truck is your XML document.

The SOAP platform and language independence makes Web services great choices for creating interfaces between disparate systems. For example, a Linux-based application can offer data access via a Web service that's consumed by an ASP.NET Web application.

ASP.NET takes care of writing SOAP requests and responses and creating the XML messages for you when working with simple requests. Complex requests will require that you know how to work with SOAP. You don't need to learn any new technical skills to start using Web services right away. To see an example of the SOAP responses and requests generated by ASP.NET, just run your Web service. Figure 6-8 shows an example of the SOAP messages generated for the `HelloWorld` method of the `HelloWebService`.

Web services aren't limited to using just SOAP messages. They can also use `GET` and `POST` requests, as explained in the earlier section "Testing Web services." `GET` and `POST` requests are part of the HTTP protocol. ASP.NET uses `POST` or `SOAP`, depending on which protocol it thinks works best. To ensure that your messages are always sent by using SOAP, add the following code to your Web service's `web.config` file:

```
<webServices>
    <protocols>
        <remove name="HttpPost" />
        <remove name="HttpGet" />
    </protocols>
</webServices>
```

Finding Web services

You might not always be consuming your own Web services. On the Web, you can find plenty of Web services. Web services, including the ones you create with ASP.NET, use two technologies to announce themselves to the world:

- ✦ **WSDL:** Web Services Description Language
- ✦ **UDDI:** Universal Discovery Description and Integration

WSDL is an XML document that describes the Web service, whereas UDDI is like the Web services yellow pages. You don't have to list your Web service with a UDDI directory, but you can if you want.

The WSDL tells the world (or another developer, at least) what's needed to know to use your Web service. Among other things, the WSDL provides information about the following elements:

- **Namespaces:** List the services available in the namespace. The namespace encapsulates the Web service's details by using the tags shown in the following:

  ```
  <wsdl:definitions targetNamespace=
     "http://mycompany.com/webservices">
  <wsdl:documentation>A web service that returns a Hello
     World message</wsdl:documentation>
  </wsdl:definitions>
  ```

- **Operations:** Use the `<portType>` element to define the Web methods available in the service. The `<message>` element defines the Web `methods` parameters:

  ```
  <wsdl:portType name="HelloWebServiceHttpGet">
     <wsdl:operation name="HelloWorld">
     <wsdl:documentation>
  Returns a Hello message using the name supplied,
     otherwise Hello World if name is null.
  </wsdl:documentation>
  <wsdl:input message="tns:HelloWorldHttpGetIn"/>
  <wsdl:output message="tns:HelloWorldHttpGetOut"/>
  </wsdl:operation>
  </wsdl:portType>
  <wsdl:message name="HelloWorldHttpGetIn">
  <wsdl:part name="name" type="s:string"/>
  </wsdl:message>
     <wsdl:message name="HelloWorldHttpGetOut">
  <wsdl:part name="Body" element="tns:string"/>
  </wsdl:message>
  ```

- **Types:** Identify the data types used by the Web service with the `<types>` element, as shown in the following:

  ```
  <wsdl:types>
     <s:schema elementFormDefault=
     "qualified" targetNamespace="http://mycompany.
     com/webservices">
     <s:element name="HelloWorld">
     <s:complexType>
     <s:sequence>
  <s:element minOccurs="0" maxOccurs="1" name="name"
     type="s:string"/>
  </s:sequence>
  </s:complexType>
  </s:element>
     <s:element name="HelloWorldResponse">
     <s:complexType>
     <s:sequence>
  ```

```
<s:element minOccurs="0" maxOccurs="1"
   name="HelloWorldResult" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="string" nillable="true" type=
   "s:string"/>
</s:schema>
</wsdl:types>
```

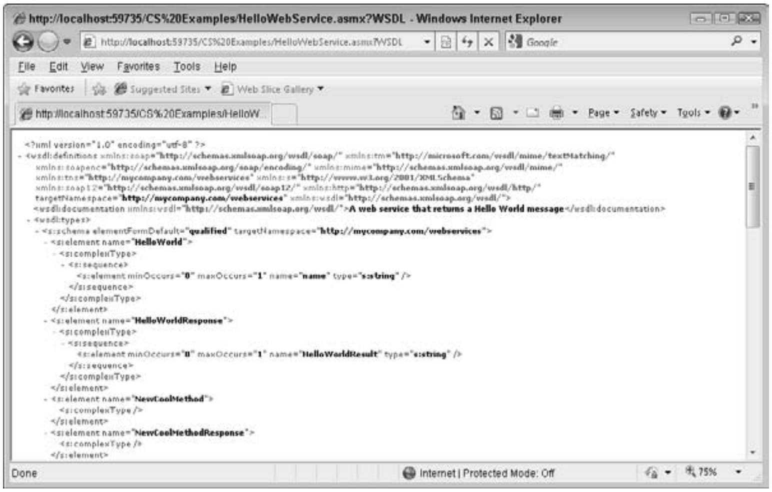
- **Protocols:** Use the `<binding>` element to define the message format, such as SOAP, and to set details for the port:

```
<wsdl:binding name="HelloWebServiceHttpGet" type="tns:
   HelloWebServiceHttpGet">
<http:binding verb="GET"/>
   <wsdl:operation name="HelloWorld">
<http:operation location="/HelloWorld"/>
   <wsdl:input>
<http:urlEncoded/>
</wsdl:input>
   <wsdl:output>
<mime:mimeXml part="Body"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
```

- **Services:** Identify the collection of protocols available for use with the service:

```
<wsdl:service name="HelloWebService">
<wsdl:documentation>A web service that returns a Hello
   World message</wsdl:documentation>
   <wsdl:port name="HelloWebServiceSoap" binding="tns:
   HelloWebServiceSoap">
<soap:address location="http://localhost:
   4251/HelloWebService/HelloWebService.asmx"/>
</wsdl:port>
   <wsdl:port name="HelloWebServiceSoap12" binding="tns:
   HelloWebServiceSoap12">
<soap12:address location="http://localhost:4251/
   HelloWebService/HelloWebService.asmx"/>
</wsdl:port>
   <wsdl:port name="HelloWebServiceHttpGet" binding="tn
   s:HelloWebServiceHttpGet">
<http:address location="http://localhost:4251/
   HelloWebService/HelloWebService.asmx"/>
</wsdl:port>
</wsdl:service>
```

ASP.NET automatically creates a WSDL document for your Web service. To view the WSDL for your Web service, as shown in Figure 6-10, click Service Description on your Web service's test page.



http://localhost:59735/CS%20Examples/HelloWebService.asmx?WSDL - Windows Internet Explorer
http://localhost:59735/CS%20Examples/HelloWebService.asmx?WSDL
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
http://localhost:59735/CS%20Examples/HelloW...
Page
Safety
Tools
Done
Internet | Protected Mode: Off
75%

```
<?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
   xmlns:tns="http://mycompany.com/webservices" xmlns:s="http://www.w3.org/2001/XMLSchema"
   xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
   targetNamespace="http://mycompany.com/webservices" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">A web service that returns a Hello World message</wsdl:documentation>
 - <wsdl:types>
   - <s:schema elementFormDefault="qualified" targetNamespace="http://mycompany.com/webservices">
     - <s:element name="HelloWorld">
       - <s:complexType>
         - <s:sequence>
             <s:element minOccurs="0" maxOccurs="1" name="name" type="s:string" />
           </s:sequence>
         </s:complexType>
       </s:element>
     - <s:element name="HelloWorldResponse">
       - <s:complexType>
         - <s:sequence>
             <s:element minOccurs="0" maxOccurs="1" name="HelloWorldResult" type="s:string" />
           </s:sequence>
         </s:complexType>
       </s:element>
     - <s:element name="NewCoolMethod">
         <s:complexType />
       </s:element>
     - <s:element name="NewCoolMethodResponse">
         <s:complexType />
       </s:element>
```

Figure 6-10: Click Service Description to view the WSDL document for your Web service.

UDDI provides a directory listing for your Web service. These are the three UDDI directories:

- **Green Pages:** Lists services offered by the registrant
- **White Pages:** Lists basic address information
- **Yellow Pages:** Categorizes businesses based on standard industrial classifications

TIP

Many companies set up UDDI directories within their organizations for internal consumption only. This allows developers and other consumers of Web services to easily see what services are available. Using Web services within your organization is a great way to reuse code.

You can access a public UDDI directory at

```
http://soapclient.com/uddisearch.html
```

Using a Web service in your application

Visual Studio provides excellent support for using a Web service in your application. It doesn't matter what language was used to create the Web service. When you reference the Web service in your application, Visual Studio places a wrapper around the Web service that allows you to access the Web service like it was your own code.

Adding a Web reference

To access a Web service in your projects, you must add a Web reference. When you add a Web reference, Visual Studio generates code — a *proxy class* — that you can use to access the Web reference.

To add a Web reference to a Web service, follow these steps:

1. **Right-click the project and choose Add Web Reference.**

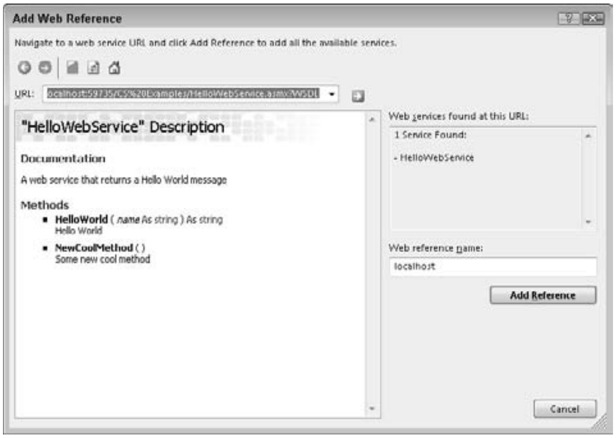
   The Add Web Reference window appears.

2. **Type or paste the URL for a Web service in the URL text box.**
3. **Click the Go button.**

   The Web service appears in the window, as shown in Figure 6-11.

4. **Click the Add Reference button.**

   The reference appears in Solution Explorer.



Add Web Reference

Navigate to a web service URL and click Add Reference to add all the available services.

URL: localhost:59735/CS%20Examples/HelloWebService.asmx?WSDL

"HelloWebService" Description

Documentation

A web service that returns a Hello World message

Methods

- **HelloWorld** (*name* As string) As string
  Hello World
- **NewCoolMethod** ( )
  Some new cool method

Web services found at this URL:

1 Service Found:

- HelloWebService

Web reference name: localhost

Figure 6-11: Use the Add Web Reference window to add a Web service to your project.

Binding with BindingSource

The `BindingSource` component is the preferred way to data-bind in Windows applications. You can use a Web service as a data source for a `BindingSource` component.

Here's how to use a Web service with a `BindingSource`:

1. **Add a Web Reference to your Web service in your Windows project.**

   If you're working with a Windows Form application, add the Web reference as a Service Reference by right-clicking the solution and choosing Add Service Reference. The process works precisely the same as the Web reference discussed earlier.

2. **Create a new instance of your Web service.**

   For example, to create a new instance of the HelloWebService, use the following in a Web application:

   ```
   private localhost.HelloWebService ws = new localhost.HelloWebService();
   ```

   When working with a desktop application, use this following code instead:

   ```
   private localhost.HelloWebServiceSoapClient ws = new localhost.
     HelloWebServiceSoapClient();
   ```

3. **Drag and drop a `BindingSource` component from the Data tab of the Toolbox onto your Windows Form.**

4. **Add your Web service as a data source for the `BindingSource` component by using the component's `Add` method.**

   For example, the following code adds HelloWebService to a BindingSource component:

   ```
   this.bindingSource1.Add(ws.HelloWorld(this.txtEnterName.Text));
   ```

5. **Set the `BindingSource` component as the source for a control's data binding.**

   For example, the following code data binds the BindingSource component to a label control's text property:

   ```
   this.lblDisplayMessage.DataBindings.Add("Text", this.bindingSource1, "");
   ```

You can use the preceding code in a button's Click event to call the Web service. Listing 6-1 shows the entire code sample in a button's Click event.

Listing 6-1: Calling a Web Service in a Button's Click Event

```
private void button1_Click(object sender, EventArgs e)
{
            this.bindingSource1.Add(ws.HelloWorld(this.txtEnterName.Text));
            this.lblDisplayMessage.DataBindings.Add("Text", this.bindingSource1,
    "");
}
```

When a user clicks the button, the code calls the HelloWorld method of the HelloWebService. The name parameter is passed by using the value entered by a user in the txtEnterName text box. The return value appears in a label control called lblDisplayMessage.

Putting your Web service in a Web application

You can use Web services in a Web application. Because ASP.NET creates a proxy class, you can call your Web service just like you'd call any other bit of code.

To use a Web service in a Web application, follow these steps:

1. **Add a Web reference to the Web service you wish to consume.**
2. **Create an instance of your Web service:**

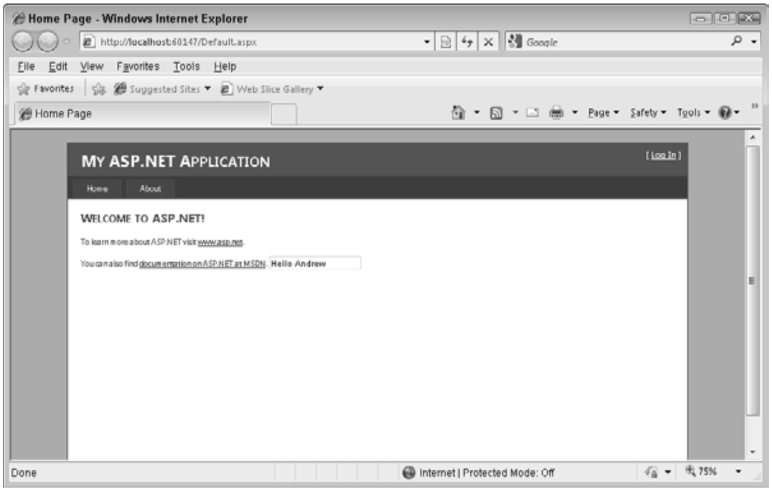
   ```
   localhost.HelloWebService ws = new localhost.HelloWebService();
   ```

3. **Call your Web service.**

   For example, the following code calls the `HelloWebService` and returns the results to a text box:

   ```
   this.TextBox1.Text = ws.HelloWorld("Andrew");
   ```

You can see the results of this example in Figure 6-12.



MY ASP.NET APPLICATION

WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

You can also find documentation on ASP.NET at MSDN. Hello Andrew

Home Page - Windows Internet Explorer
http://localhost:60147/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page Safety Tools
[Log In]
Home
About
Done
Internet | Protected Mode: Off
75%

Figure 6-12: Consuming a Web service in a Web application.

Getting on the client side

You aren't limited to just accessing Web services via server-side code. You can also use client-side scripting languages, such as JScript, to call Web services.

Web services return an XML document, for which JScript provides excellent support. Listing 6-2 is a simple script and HTML page that tests the `HelloWebService`.

Listing 6-2: Simple Script and HTML Page That Tests the `HelloWebService`

```
<html>
<head>
    <title>Test Web Service</title>
    <script type="text/jscript">
    function callService()
    {
        var xmlDoc;
        var url;

         xmlDoc = new ActiveXObject
   ("Msxml2.DOMDocument");
         xmlDoc.onreadystatechange = function ()
             {
                if (xmlDoc.readyState == 4)
                     {
                        displayText(xmlDoc);
                    }
            }

        url =
           "http://<servername>/HelloWebService/" +
           "HelloWebService.asmx/HelloWorld?name=" +
           document.getElementById("text1").value;

        xmlDoc.load(url);

    }

    function displayText(xmlDoc)
    {

         var currentNode;
         currentNode = xmlDoc.selectSingleNode("string");

         document.getElementById("displayMessage").
   innerHTML = currentNode.text;

    }

    </script>
</head>
<body>
    <form>
        Enter your name:
        <input id="Text1" type="text" name="name" />
        <input id="Button1" type="button" value="button"
   onclick="callService()" />
        <div id="displayMessage"></div>
    </form>
</body>
</html>
```

The script works only in Internet Explorer because it uses an ActiveX control, which other browsers don't support. You can find lots of scripts on the Web for manipulating XML with JScript.

You should definitely take a closer look at the `XMLHTTPRequest` object. This object allows you to send `GET` and `POST` requests to a Web server. Best of all, it works without causing the browser to refresh. The `XMLHTTPRequest` object is at the heart of the Web development methodology called AJAX.

Chapter 7: Handling Exceptions and Debugging

In This Chapter

- **Using `Try...Catch...Finally` blocks**
- **Throwing exceptions**
- **Viewing exceptions in Exception Assistant**
- **Setting breakpoints and stepping through code**
- **Using the Watch window to view data**
- **Debugging code generated by Visual Studio**

Anyone who has done even a little coding has been deflated by seeing the Visual Studio Debugger open instead of a beautiful masterpiece — because of an error that occurs in the program code. Sometimes, you spend more time with the Visual Studio Debugger than you do writing code. Of course, you might see less of the Debugger if you start using structured exception handling to capture exceptions when they do occur. And this chapter shows you how to do just that.

This chapter provides you with a full view of the Debugger. For example, you discover the wonders of using breakpoints to control when Visual Studio pauses your code and lets you debug it. You can also read how the Watch window works. All these tools can make your debugging significantly easier.

Structured Exception Handling to the Rescue

Whether you're writing code as a professional or a hobbyist, you want your code to perform as advertised. For example, when a user clicks a button to load information from a file, you want to make sure that file is there. If the file isn't there, you need to have some way to deal with the error.

In object-oriented programming (OOP), errors are often called *exceptions*. An exception is just what it sounds like — anything that occurs in your code that's exceptional or out of the ordinary. For example, if your code opens a file and the file server goes down while you're accessing the file, that's an exception. Comparatively, not being able to access the file server or find a file in the first place isn't an exception because your code can test to see whether the file exists before it tries to open the file.

Code you write to address an exception is an *exception handler.* Exceptions that occur for which you haven't written code are *unhandled exceptions,* which can be fatal and cause your application to fail. Not good.

Visual Studio doesn't provide exception-handling features. In many cases, you'll use the built-in exception handlers supplied with the .NET Framework. In some cases, you deal with exceptions by writing exception handlers in the language of your choice. You can use Visual Studio to do the following:

- Step through error-producing code in the Visual Studio Debugger.
- Use Exception Assistant to view details about an exception.
- Use code snippets to add exception handlers to your code.

The .NET Framework provides a model for managing exceptions that includes

- Structured exception handling
- A common-exception framework
- The ability to create your own custom exceptions

Handling exceptions

In the good old days (say, more than six years ago), it was every man for himself when it came to handling exceptions. You could return a value when an exception occurred or do nothing at all. Although plenty of ways are available with which you *can* handle exceptions, you *should* handle exceptions by using structured exception handling. Structured exception handling makes it possible to handle exceptions in a predictable manner. Structured exception handling also reduces the learning curve for novice developers who must learn exception handling techniques and makes your code easier to read. Consequently, everyone wins when you use structured exception handling techniques.

Structured exception handling involves using the `Try`, `Catch`, and `Finally` statements to create code blocks:

- `Try`: Executes statements that might cause an exception, such as opening database connections

- `Catch`: Specifies the exception to catch and executes code to deal with the exception
- `Finally`: Executes every time, whether exceptions generate or not

Another important feature of structured exception handling is the `throw` statement, which you use to create a new exception or to pass an existing exception to the calling function.

Here's an example of a `Try...Catch...Finally` block for opening and closing a database connection with ADO.NET:

```
Private Sub AccessData()
  Dim MyConnection As New System.Data.SqlClient.SqlConnection
  MyConnection.ConnectionString = My.Settings.MyDbString
  Try
     MyConnection.Open()
     'send commands
  Catch ex As Exception
     'handle exception
  Finally
     MyConnection.Close()
  End Try
End Sub
```

The statements executed in the `Try` block are wired to the `Catch` block. If any statement in the `Try` block throws an exception, the `Catch` block gives you a chance to

- Attempt to recover from the exception.
- Log the exception.
- Provide feedback to the user.

Any code in the optional `Finally` block executes, regardless of whether an exception occurred. Place your cleanup code, such as closing database connections, in the `Finally` block.

Catching exceptions

The .NET Framework provides an extensive catalog of exceptions that starts with the generic `System.Exception`. All exceptions inherit from `System.Exception`. You use a `Catch` block to capture exceptions and handle them. Table 7-1 lists examples of exceptions found in the .NET Framework.

Table 7-1 Example Exceptions Found in the .NET Framework

Exception	*Usage*
`System.Exception`	A generic exception
`System.NullReferenceException`	Occurs when you attempt to access the value of an object that doesn't exist
`System.ArgumentNullReferenceException`	Occurs when an unexpected `nullreference` is passed as an argument
`System.Data.SqlClient.SqlException`	An exception thrown when SQL Server returns a warning or error
`System.IndexOutofRangeException`	Thrown when you attempt to access beyond an array's index
`System.InvalidCastException`	Thrown when an attempt to cast from one data type to another is invalid

You can handle any of the exceptions listed in Table 7-1 by using a `Catch` block.

`Catch` blocks use the following syntax:

- The keyword `Catch`
- An optional argument that specifies the kind of exception to catch

Here is a `Catch` block that catches an `SQLException`:

```
Catch ex As System.Data.SqlClient.SqlException
```

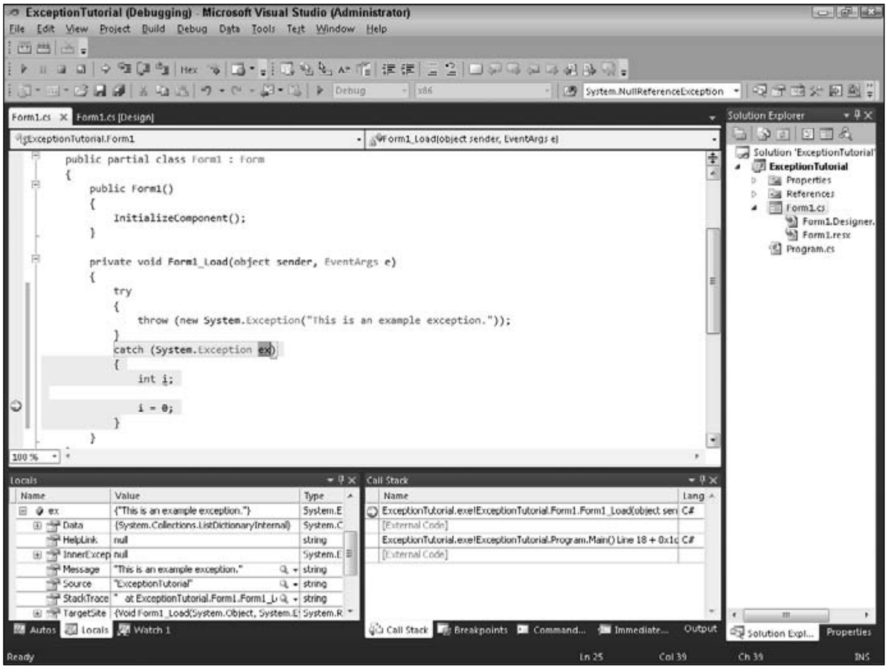
You can use multiple `Catch` blocks to catch different kinds of exceptions. Depending on your application, you may want to catch the most specific exception first. The least specific exception is `System.Exception`.

For example, here are two `Catch` blocks:

```
Catch ex As System.Data.SqlClient.SqlException
   'do something with the exception

Catch ex As System.Exception
   'do something with the exception
```

Only one of these `Catch` blocks executes. Always list your `Catch` blocks in order of most specific to least specific. While your code executes, it starts at the first `Catch` block and steps through each until it finds a match. If `System.Exception` is the first `Catch` block, no other `Catch` blocks execute.

Figuring out which exceptions you should use can be difficult. One option is to add the exception to the Watch window while debugging. That allows you to see the specific exception captured. Figure 7-1 shows an example of a `System.Exception` caught in the Watch window. See the section "Understanding Debugging," later in this chapter, for more information about using Watch windows.



```
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            try
            {
                throw (new System.Exception("This is an example exception."));
            }
            catch (System.Exception ex)
            {
                int i;

                i = 0;
            }
        }
```

Locals

Name	Value	Type
ex	{"This is an example exception."}	System.E
Data	{System.Collections.ListDictionaryInternal}	System.C
HelpLink	null	string
InnerExcep	null	System.E
Message	"This is an example exception."	string
Source	"ExceptionTutorial"	string
StackTrace	" at ExceptionTutorial.Form1.Form1_L	string
TargetSite	{Void Form1_Load(System.Object, System.E	System.R

Call Stack

Name	Lang
ExceptionTutorial.exe!ExceptionTutorial.Form1.Form1_Load(object sen	C#
[External Code]	
ExceptionTutorial.exe!ExceptionTutorial.Program.Main() Line 18 + 0x1c	C#
[External Code]	

Figure 7-1: Use the Watch window to puzzle out information about exceptions.

As Figure 7-1 shows, exceptions provide a lot of information. Some useful properties of exceptions include

- `Data`: Gets a set of key/value pairs that provides optional, additional information about the exception
- `GetBaseException`: Gets the root exception — the *base exception* — that causes a chain of exceptions to start

- InnerException: Gets the exception that causes the current exception

  The base exception's InnerException is null because the base exception is the first exception in the chain.

- Message: Gets the exception's description
- Source: Is used to get or set the name of the application or object that causes the exception

  You can set the Source property before you pass an exception to another method to process.

- StackTrace: Provides a list of all the methods called before the exception occurs
- TargetSite: Gets the method that causes the exception

  The method provided by the TargetSite is the same method listed at the top of the StackTrace.

Always capture information about exceptions. Exceptions don't always occur while you're sitting in front of your laptop with a cup o' Joe and Debugger fired up. Exceptions occur when you're two weeks past your deadline on a new project and the last thing you have time for is something you haven't thought about in six months to start malfunctioning. You can use an exception's properties to puzzle out what caused the exception.

Throwing exceptions

When an exception occurs, the exception is *thrown.* The source of this term is the keyword throw, which is used to raise an exception when an error occurs.

Here are the two uses of the throw statement:

- **To raise an exception in your program:** The syntax to throw an exception is throw someException.

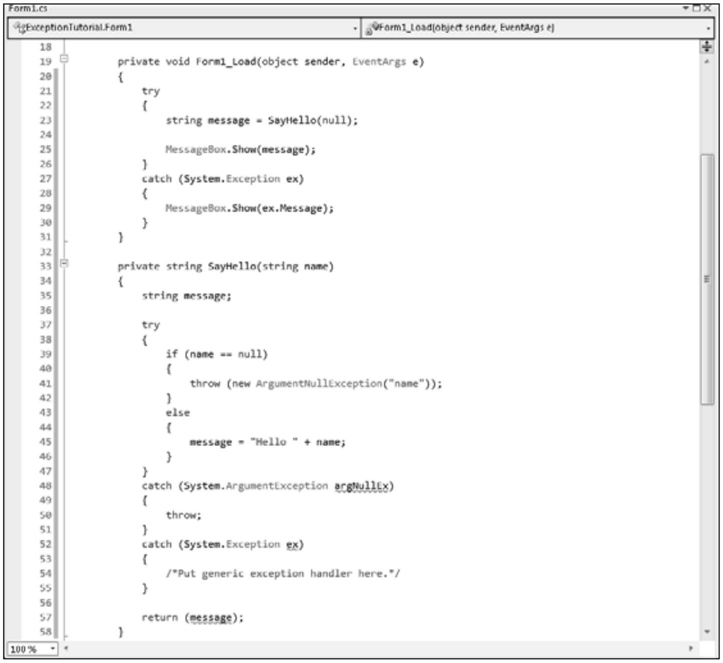
  For example, to throw an exception when a null value is passed to a method, use System.ArgumentNullException, as the following code demonstrates:

```
private string SayHello(string name)
{
   if (name == null)
   {
      throw (new ArgumentNullException("name"));
   }
   else
   {
      return ("Hello " + name);
   }
}
```

- **To pass an exception to a calling function:** You use the `throw` statement within a `Catch` block to pass an existing exception to a calling function, as the following code shows:

```
try
{
    /*Do something that causes an exception*/
}
catch (System.Exception ex)
{
}
```

The `Catch` block captures the exception, and the `throw` statement passes the captured exception to the calling function. Figure 7-2 shows an example of the `throw` statement in action.



```
18
19          private void Form1_Load(object sender, EventArgs e)
20          {
21              try
22              {
23                  string message = SayHello(null);
24
25                  MessageBox.Show(message);
26              }
27              catch (System.Exception ex)
28              {
29                  MessageBox.Show(ex.Message);
30              }
31          }
32
33          private string SayHello(string name)
34          {
35              string message;
36
37              try
38              {
39                  if (name == null)
40                  {
41                      throw (new ArgumentNullException("name"));
42                  }
43                  else
44                  {
45                      message = "Hello " + name;
46                  }
47              }
48              catch (System.ArgumentException argNullEx)
49              {
50                  throw;
51              }
52              catch (System.Exception ex)
53              {
54                  /*Put generic exception handler here.*/
55              }
56
57              return (message);
58          }
```

Figure 7-2: Use a `throw` statement to pass an exception message to a calling function.

Note the following in the code sample shown in Figure 7-2:

- The subroutine `Form1_Load()` calls the function `SayHello()` on line 23.
- On line 41, `SayHello()` throws an exception of the type `System.ArgumentNullException` because a `null` value was passed in as a parameter.

- Code execution jumps to line 48, where the `Catch` block captures the exception.
- Line 50 throws the exception to the caller, `Form1_Load()`.
- Execution returns to line 27, the `Catch` block in `Form1_Load()`, which captures the exception thrown to it by line 50.
- Line 29 displays a message box that displays the exception's `Message` property, which is `Value cannot be null`.

Other times when you might want to use the `throw` statement include

- **To test your exception handlers:** You may find it helpful to have your unit test or other testing harness throw exceptions to test your exception handlers and logging utilities.
- **When you use an exception-handling framework:** You should handle exceptions for many patterns. For example, in ASP.NET Web sites, common exception handlers are typically placed in the `Globals.aspx` page. All your code's exception handlers simply use the `throw` statement. Any exceptions are automatically handled by an application-wide handler in `Globals.aspx`.

TIP

Using a centralized exception handler makes it easier to ensure that all your exceptions are logged properly. You can also make sure your users are sent to a consistent error page.

Using Visual Studio to manage exceptions

The Microsoft .NET Framework provides an extensive set of exceptions that you can catch and throw. Although you can create your own exceptions, that isn't necessary in most cases. Instead, use Object Browser to view the list of exceptions available in the .NET Framework.

To use Object Browser, follow these steps:

1. **Press Ctrl+Alt+J to open Object Browser.**
2. **From the Browse drop-down list, choose .NET Framework 4.**
3. **Type** exception **in the Search box and then press Enter.**

   A list of objects with the word *exception* in their names appears.

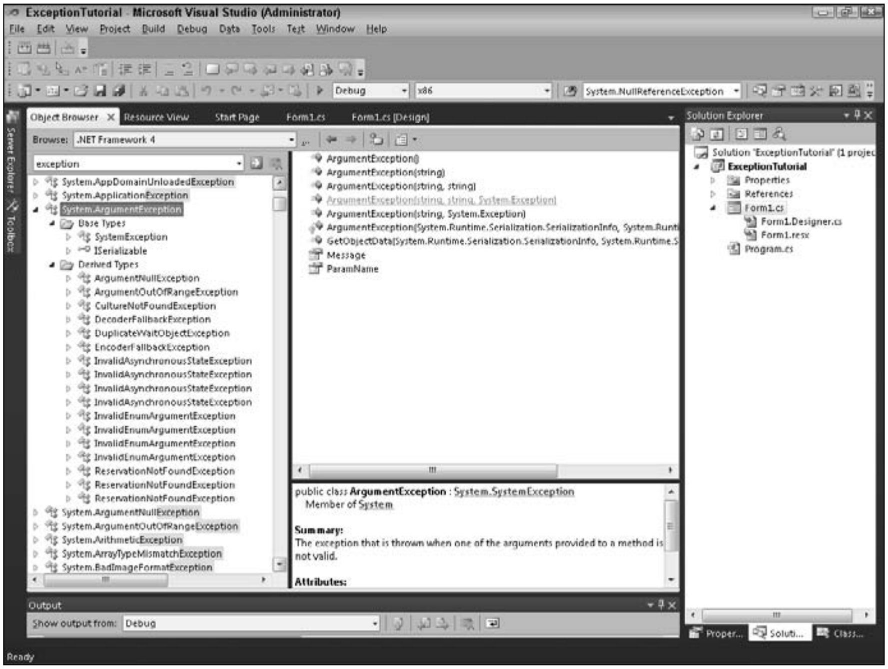
   The .NET Framework uses a naming standard whereby all exceptions end with the word `exception`, such as `System.FormatException`.
4. **Right-click one of the entries listed in Object Browser and choose Sort by Object Type from the shortcut menu that appears.**

   The objects are sorted by type. That way, you see all the exception objects grouped in the Classes folder.

By default, the list of objects is sorted alphabetically.

To view more information about an exception, click the plus sign (+) to expand the exception. Click an exception's name to view its properties and methods. A summary pane lists summary information about the exception. Figure 7-3 shows the exceptions derived from the `System.ArgumentException` object.

To read more about an exception, click the exception in Object Browser and then press F1 to display help about the exception.



Debug x86 System.NullReferenceException

Browse: .NET Framework 4

exception

```
▷ System.AppDomainUnloadedException
▷ System.ApplicationException
◢ System.ArgumentException
    ◢ Base Types
        ▷ SystemException
        ▷ ISerializable
    ◢ Derived Types
        ▷ ArgumentNullException
        ▷ ArgumentOutOfRangeException
        ▷ CultureNotFoundException
        ▷ DecoderFallbackException
        ▷ DuplicateWaitObjectException
        ▷ EncoderFallbackException
        ▷ InvalidAsynchronousStateException
        ▷ InvalidAsynchronousStateException
        ▷ InvalidAsynchronousStateException
        ▷ InvalidAsynchronousStateException
        ▷ InvalidEnumArgumentException
        ▷ InvalidEnumArgumentException
        ▷ InvalidEnumArgumentException
        ▷ InvalidEnumArgumentException
        ▷ ReservationNotFoundException
        ▷ ReservationNotFoundException
        ▷ ReservationNotFoundException
▷ System.ArgumentNullException
▷ System.ArgumentOutOfRangeException
▷ System.ArithmeticException
▷ System.ArrayTypeMismatchException
▷ System.BadImageFormatException
```

```
ArgumentException()
ArgumentException(string)
ArgumentException(string, string)
ArgumentException(string, string, System.Exception)
ArgumentException(string, System.Exception)
ArgumentException(System.Runtime.Serialization.SerializationInfo, System.Runti
GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.S
Message
ParamName
```

public class **ArgumentException** : System.SystemException
Member of System

Summary:
The exception that is thrown when one of the arguments provided to a method is not valid.

Attributes:

```
Solution 'ExceptionTutorial' (1 projec
◢ ExceptionTutorial
    ▷ Properties
    ▷ References
    ◢ Form1.cs
        Form1.Designer.cs
        Form1.resx
    Program.cs
```

Show output from: Debug

Figure 7-3: Use Object Browser to explore the exceptions available in the .NET Framework.

TIP

In most cases, you should use the most specific exception type available. For example, `System.DivideByZeroException` is more specific than `System.ArithmeticException`. In most cases, you can tell which exception is more specific by looking at the name. An arithmetic exception will include divide-by-zero errors. You can also look at the Derived Types list, as shown in Figure 7-3. Exceptions that appear in this list are based on a more generic exception.

REMEMBER

All exceptions inherit from `System.Exception`. As a result, all exceptions have the same properties of `System.Exception`.

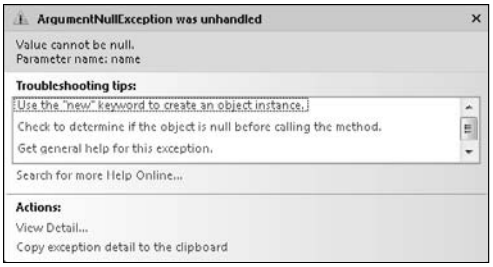
Visual Studio provides Exception Assistant for viewing information about exceptions while using Visual Studio Debugger. Visual Studio displays Exception Assistant any time exceptions are either of the following conditions:

- **Unhandled:** Any time Visual Studio encounters an exception while executing your code, Exception Assistant appears.
- **Caught:** While using the Visual Studio Debugger, you can open Exception Assistant if the Debugger breaks at the point where the exception is caught. Press Shift+Alt+F10 to open Exception Assistant.

Figure 7-4 shows an example of Exception Assistant for an unhandled exception. Exception Assistant provides you with

- **The exception's message**
- **Tips for troubleshooting the exception message**
- **The exception's properties**

  Click the View Detail link in the Actions portion of the Exception Assistant dialog box, as shown in Figure 7-4, to view the exception's properties.



ArgumentNullException was unhandled

Value cannot be null.
Parameter name: name

Troubleshooting tips:

Use the "new" keyword to create an object instance.

Check to determine if the object is null before calling the method.

Get general help for this exception.

Search for more Help Online...

Actions:

View Detail...

Copy exception detail to the clipboard

Figure 7-4: Exception Assistant appears when an unhandled exception is encountered.

Understanding Debugging

When you use debugging, you can stop your code during execution and inspect it. Visual Studio provides a sophisticated Debugger that allows you to

- Control the execution of your code so that you can peek inside your code while it's running.
- Test your code while you're designing it.

- View the data used by your code.
- Get detailed information about exceptions that occur.

Enabling debugging

To use the Visual Studio Debugger with your code, you must build your code by using the Debug build configuration. The Debug build configuration tells the compiler to create a program database (PDB) file for your code. The PDB file stores the data about your source code, such as

- Source code line numbers
- Variable names
- Method names

The Visual Studio Debugger uses the PDB file to know how to access your source code while it's being executed.

The data stored in the PDB are *symbols.*

You should generate PDB files for all builds, including release builds, so that you can debug the build, even if you didn't build it as a debug build.

To set your application to use the Debug build configuration, choose Debug from the Solution Configurations drop-down list on the Standard toolbar. To create or modify build configurations, see Book VI, Chapter 1.

Firing up the Debugger

Your application has two modes in the Debugger: executing or breaking. While your application is executing, everything's A-okay. After your application stops executing, or breaks, then you get busy with the Debugger.

Your application executes until it encounters either of the following:

- **Breakpoint:** A marker you set in the code that tells the Debugger to stop execution
- **Exception:** An unhandled error generated by your code that causes program execution to stop

While your code is in break mode, you have the features of the Debugger at your disposal for examining your code.

Controlling execution

One of the primary features of the Visual Studio Debugger is the ability to control your program's execution. Controlling execution allows you to walk through your code, as well as check the values of variables and data.

The Debugger provides the following code execution options:

- **Starting:** This starts your code. You must use the Debug build configuration for the Debugger to fire up.
- **Stepping:** This allows you to step through your code.
- **Breaking:** You can break the code's execution by setting a breakpoint or by manually breaking. Your code breaks automatically when an unhandled exception occurs.
- **Stopping:** Execution stops when your program completes execution or you manually stop it.

You have several ways to start the Debugger:

- **Debug menu:** From here, choose Start Debug, Step Into, or Step Over to execute your code.
- **Run to Cursor:** Right-click an executable line of code and choose Run to Cursor from the shortcut menu. The Debugger starts your code and breaks at the line where your cursor sits.
- **Keyboard shortcuts:** Table 7-2 lists common keyboard shortcuts you can use to control the Debugger.

  You can use the keyboard shortcuts almost exclusively to step through code.
- **Debug toolbar:** The Debug toolbar acts like standard VCR buttons for controlling your code's execution. Figure 7-5 shows the Debug toolbar.

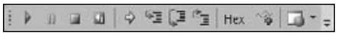


Figure 7-5: Use the Debug toolbar to control your code's execution.

Table 7-2 Common Debugging Keyboard Shortcuts

Keyboard Shortcut	*Command*
F5	Start Debugging
Ctrl+F5	Start without Debugging
F11	Step Into
F10	Step Over
Shift+F11	Step Out

Breaking execution

The first step in using the Debugger is breaking your code during execution. To break your code during execution, you can do any one of the following:

- Click the Break All button on the Debug toolbar.
- Choose Break All from the Debug menu.
- Press Ctrl+Alt+Break.

In most cases, you'll likely set a breakpoint in your code before you start execution. By setting a breakpoint, your code stops executing at the line of code where the breakpoint is set. You can then step through the code and watch your variables while the code executes.

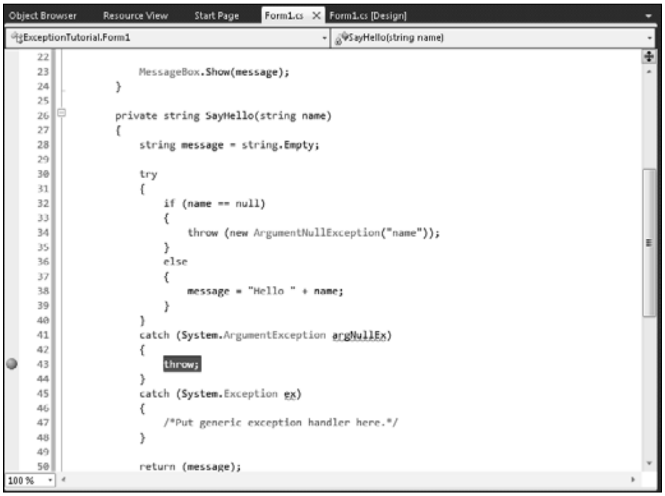
Here's how to set a breakpoint:

1. **Go to the line of source code in the Code Editor where you want code execution to break.**
2. **From the Debug menu, choose Toggle Breakpoint.**

   A solid glyph (red circle) appears in the Code Editor's left gutter. The breakpoint appears in the Breakpoints window. Figure 7-6 shows the breakpoint.

You can use alternative ways to set breakpoints, such as the following:

- Click the left gutter in the Code Editor to toggle a breakpoint for an executable line of code.
- Right-click a line of code and choose Breakpoint from the shortcut menu.
- Press F9.



```
            MessageBox.Show(message);
        }

        private string SayHello(string name)
        {
            string message = string.Empty;

            try
            {
                if (name == null)
                {
                    throw (new ArgumentNullException("name"));
                }
                else
                {
                    message = "Hello " + name;
                }
            }
            catch (System.ArgumentException argNullEx)
            {
                throw;
            }
            catch (System.Exception ex)
            {
                /*Put generic exception handler here.*/
            }

            return (message);
```

Figure 7-6: Set a breakpoint in the Code Editor.

You can use the Breakpoints window to manage your breakpoints. From here, you can

- Create new breakpoints.
- Enable or disable breakpoints.
- Set conditions on when the breakpoint occurs.
- Filter the breakpoint for specific machines, processes, and threads.
- Specify how many times the code should execute before execution breaks.

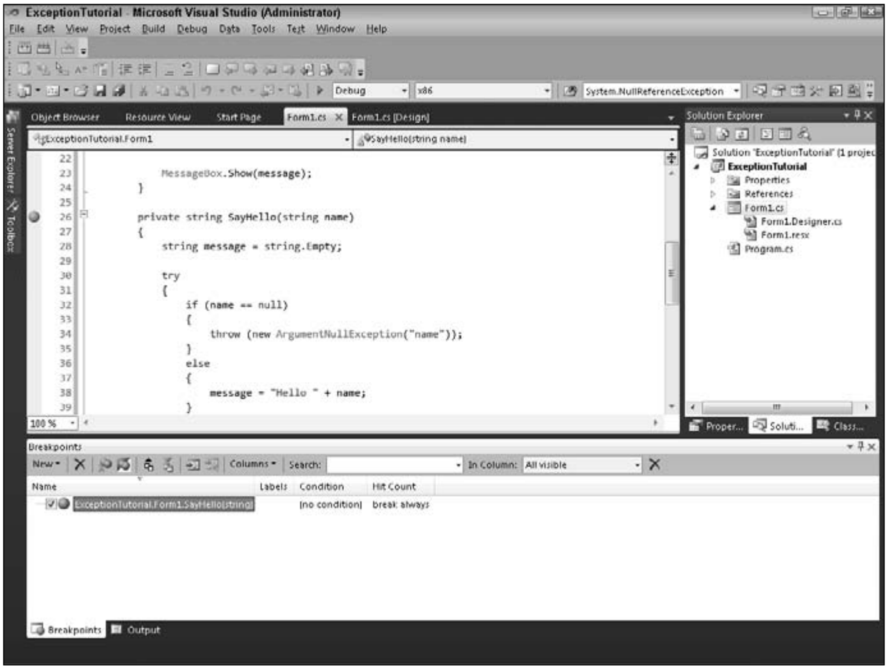
Here’s how to create a new breakpoint via the Breakpoints window:

1. **Press Ctrl+Alt+B to display the Breakpoints window.**
2. **Choose New➪Break at Function.**

   The New Breakpoint window appears.
3. **In the Function text box, type the name of the function where you want to create the breakpoint.**

   For example, type SayHello() to create a breakpoint at a function called `SayHello()`.
4. **Click OK.**

   The function appears in the Breakpoints window, as shown in Figure 7-7.



```
        MessageBox.Show(message);
    }

    private string SayHello(string name)
    {
        string message = string.Empty;

        try
        {
            if (name == null)
            {
                throw (new ArgumentNullException("name"));
            }
            else
            {
                message = "Hello " + name;
            }
```

Name	Labels	Condition	Hit Count
☑ ● ExceptionTutorial.Form1.SayHello(string)		[no condition]	break always

Figure 7-7: Use the Breakpoints window to add new breakpoints.

Stepping through code

Visual Studio provides several options for controlling the execution of your code. Besides just starting, pausing, and stopping code execution, you can step through your code one statement at a time. The three step options are

- **Step Into:** Breaks on each line of code
- **Step Over:** Breaks on each line of code except functions

  The function executes, and code execution breaks at the first line outside the function.
- **Step Out:** Resumes execution on a function and then breaks at the first line of code outside the function

Step Into and Step Over start the Debugger at the next line of code. Step Out allows you to stop debugging a function but then break execution as soon as the function exits.

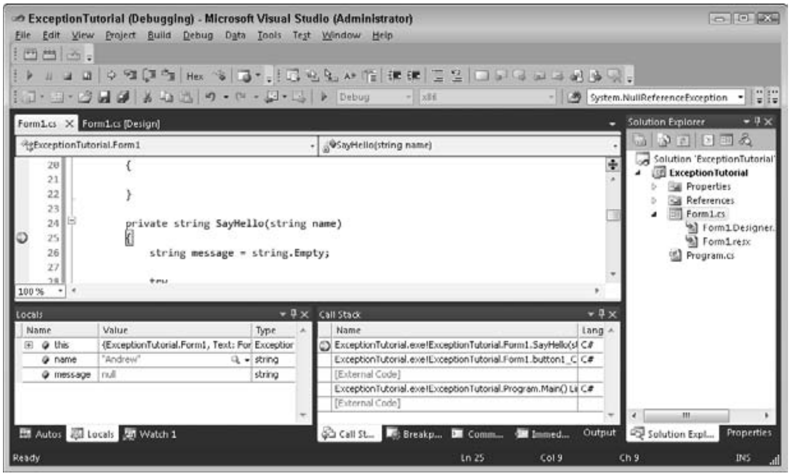
To step through your code, follow these steps:

1. **Set breakpoints in your code.**

   For example, create a form with a button and add a `Click` event for the button. Set the breakpoint on the button's `Click` event to break the code's execution when you click the button.

2. **Press F5 to start the Debugger.**

   Visual Studio builds your code and starts execution. The code executes until it encounters a breakpoint. Figure 7-8 shows an example of the Debugger hitting a breakpoint. The Code Editor highlights the code in yellow.



```
20            {
21
22            }
23
24        private string SayHello(string name)
25        {
26            string message = string.Empty;
27
28            try
```

Locals

Name	Value	Type
this	{ExceptionTutorial.Form1, Text: For	Exceptior
name	"Andrew"	string
message	null	string

Call Stack

Name	Lang
ExceptionTutorial.exe!ExceptionTutorial.Form1.SayHello(s	C#
ExceptionTutorial.exe!ExceptionTutorial.Form1.button1_C	C#
[External Code]	
ExceptionTutorial.exe!ExceptionTutorial.Program.Main() Li	C#
[External Code]	

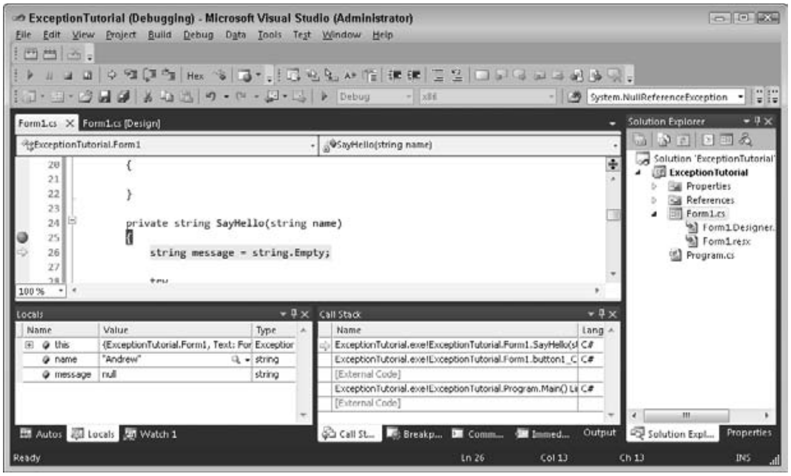
Figure 7-8: The Debugger highlights the code statement when it hits a breakpoint.

3. **Press F11 to step into the next line of execution.**
4. **Continue to press F11 until you encounter a function.**
   - *Step into:* To follow the execution into the function, press F11 to step into the function.
   - *Step over:* To execute the function without stepping through it, press F10 to step over the function.
   - *Step out:* To step out of the function after you enter it, press Shift+F11.

   Figure 7-9 shows an example of the Debugger stepping into a function. The Debugger hit the breakpoint in line 25 in Figure 7-9. Press F11 to step through each line. At line 25 is the function `SayHello()`. Step into the function, and the Debugger jumps to line 26 (see Figure 7-2), where the next statement to execute exists. The arrow in the Code Editor's left gutter shows the current line being executed.
5. **Press Shift+F11 to step out of the function.**

   The Debugger executes the function and returns to the calling line of code.

In the case of the code in Figure 7-9, pressing Shift + F11 returns you to line 53.



ExceptionTutorial (Debugging) - Microsoft Visual Studio (Administrator)

```
        {

        }

        private string SayHello(string name)
        {
            string message = string.Empty;

            try
```

Locals

Name	Value	Type
this	{ExceptionTutorial.Form1, Text: For	Exceptior
name	"Andrew"	string
message	null	string

Call Stack

Name	Lang
ExceptionTutorial.exe!ExceptionTutorial.Form1.SayHello(s	C#
ExceptionTutorial.exe!ExceptionTutorial.Form1.button1_C	C#
[External Code]	
ExceptionTutorial.exe!ExceptionTutorial.Program.Main() Li	C#
[External Code]	

Figure 7-9: Use the Debugger to step through your code.

6. **Resume stepping through the code by pressing F11.**

   F5 is the shortcut key for the Continue command. Press F5 at any time to resume execution until the code terminates or hits another breakpoint.

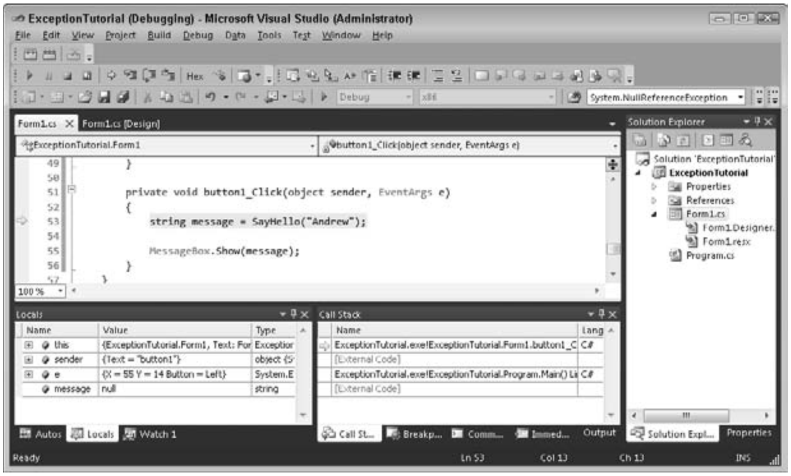
The Debug toolbar has three buttons that correspond to the Step Into, Step Over, and Step Out commands, respectively. Refer to Figure 7-5.

You can use the Run to Cursor command to execute your code to the cursor. Just right-click a line of code in the Code Editor and then choose Run to Cursor from the shortcut menu. Code execution breaks at the line where your cursor is positioned.

To know whether your code is running or debugging, look for `Running` or `Debugging`, respectively, in the title bar. Figure 7-10 shows an example.

Viewing data with the Debugger

The Visual Studio Debugger provides several windows for managing debugging. By default, these windows automatically appear in the bottom of the screen while you're debugging. You can access and open debugging windows by choosing the Debug➪Windows menu selection. Table 7-3 lists the debugging windows and their functions.



```
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string message = SayHello("Andrew");

            MessageBox.Show(message);
        }
    }
```

Name	Value	Type
this	{ExceptionTutorial.Form1, Text: For	Exceptior
sender	{Text = "button1"}	object {S
e	{X = 55 Y = 14 Button = Left}	System.E
message	null	string

Name	Lang
ExceptionTutorial.exe!ExceptionTutorial.Form1.button1_C	C#
[External Code]	
ExceptionTutorial.exe!ExceptionTutorial.Program.Main() Li	C#
[External Code]	

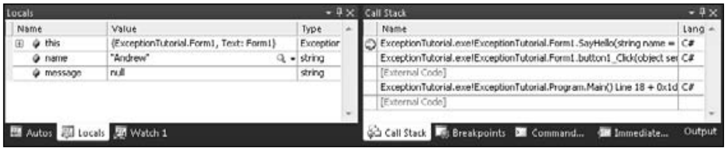
Figure 7-10: The title bar displays the execution status of your program.

Table 7-3 Debugging Windows and Their Functions

Window	*Type*	*Access*	*Shortcut Key*	*Function*
Breakpoints	Execution control	Debug menu	Ctrl+Alt+B	Creates a new breakpoint or edits an existing breakpoint
Output	Feedback	Debug menu	N/A	Displays results or feedback
Script	N/A	Debug menu	Ctrl+Alt+N	Lists scripts used by your Explorer program
Watch	Variable	Debug menu	Ctrl+Alt+W, 1	Displays variables and expressions you add to the window
Autos	Variable	Debug menu	Ctrl+Alt+V, A	Displays variables used in the current and preceding lines of code
Locals	Variable	Debug menu	Ctrl+Alt+V, L	Displays all variables in scope
Immediate	Variable	Debug menu	Ctrl+Alt+I	Executes commands and sets variable values
Call Stack	Memory	Debug menu	Ctrl+Alt+C	Displays functions in memory

Window	*Type*	*Access*	*Shortcut Key*	*Function*
Threads	CPU	Debug menu	Ctrl+Alt+H	Displays a set of instructions being executed by your program
Modules	Memory	Debug menu	Ctrl+Alt+U	Lists the software modules used by your program
Processes	Memory	Debug menu	Ctrl+Alt+Z	Displays processes which your program has launched or attached to
Memory	Memory	Debug menu	Ctrl+Alt+M, 1	Displays contents of memory
Disassembly	CPU	Debug menu	Ctrl+Alt+D	Displays assembly code of your program
Registers	CPU	Debug menu	Ctrl+Alt+G	Displays registers' contents
QuickWatch	Variable	Debug menu	Ctrl+Alt+Q	Displays values of current in break mode value or expression
Exception	Feedback	N/A	N/A	Displays information about an Assistant exception when an exception occurs
DataTips	Variable	Hover mouse over variable	N/A	Displays value of variable in scope

Figure 7-11 shows the tabs of the debugging windows at the bottom of a screen. Click a tab to access the window.



Locals

Name	Value	Type
this	{ExceptionTutorial.Form1, Text: Form1}	Exceptior
name	"Andrew"	string
message	null	string

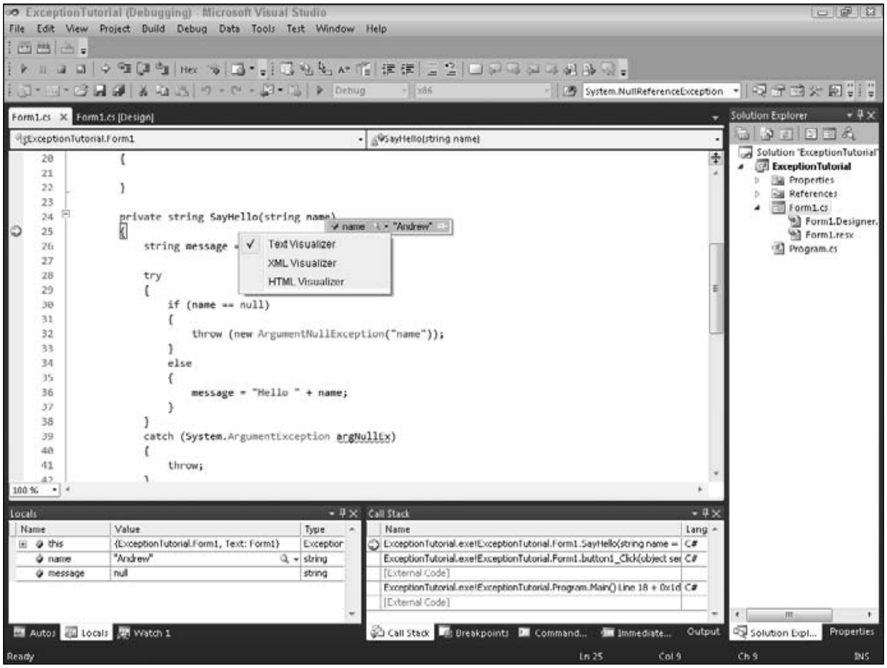
Call Stack

Name	Lang
ExceptionTutorial.exe!ExceptionTutorial.Form1.SayHello(string name =	C#
ExceptionTutorial.exe!ExceptionTutorial.Form1.button1_Click(object se	C#
[External Code]	
ExceptionTutorial.exe!ExceptionTutorial.Program.Main() Line 18 + 0x1d	C#
[External Code]	

Figure 7-11: Use the debugging windows (bottom of the screen) to debug your code.

Many of the debugging windows allow you to observe the contents of your variables and data structures.

Visual Studio offers the DataTips feature that allows you to view the contents of a variable. Simply hover your mouse over a variable that's active, and a DataTip appears that displays the variable's value. Figure 7-12 shows a DataTip. Notice that you can click the down arrow next to a DataTip and choose a different visualizer. *Visualizers* display data in different forms to make it easier to read and interpret.



```
        {

        }

        private string SayHello(string name)
        {
            string message =

            try
            {
                if (name == null)
                {
                    throw (new ArgumentNullException("name"));
                }
                else
                {
                    message = "Hello " + name;
                }
            }
            catch (System.ArgumentException argNullEx)
            {
                throw;
            }
```

Locals

Name	Value	Type
this	{ExceptionTutorial.Form1, Text: Form1}	Exceptior
name	"Andrew"	string
message	null	string

Call Stack

Name	Lang
ExceptionTutorial.exe!ExceptionTutorial.Form1.SayHello(string name =	C#
ExceptionTutorial.exe!ExceptionTutorial.Form1.button1_Click(object ser	C#
[External Code]	
ExceptionTutorial.exe!ExceptionTutorial.Program.Main() Line 18 + 0x1d	C#
[External Code]	

Figure 7-12: Hover your mouse over a variable to display its value in a DataTip.

The variable in Figure 7-12 is a `string` variable. You can use DataTips to display complex variables, such as `DataSet`s.

Visual Studio also provides the Autos, Locals, and Watch windows for viewing variable values. The Autos and Locals windows populate automatically. You use the Watch window to identify a specific variable that you want to monitor.

To use the Watch window to monitor a `DataSet`, follow these steps:

1. **Set a breakpoint in your code where you wish to monitor a variable.**
2. **Press F5 to start the Debugger.**
3. **When code execution breaks, choose Debug➪Windows➪Watch 1.**

   The Watch window appears.

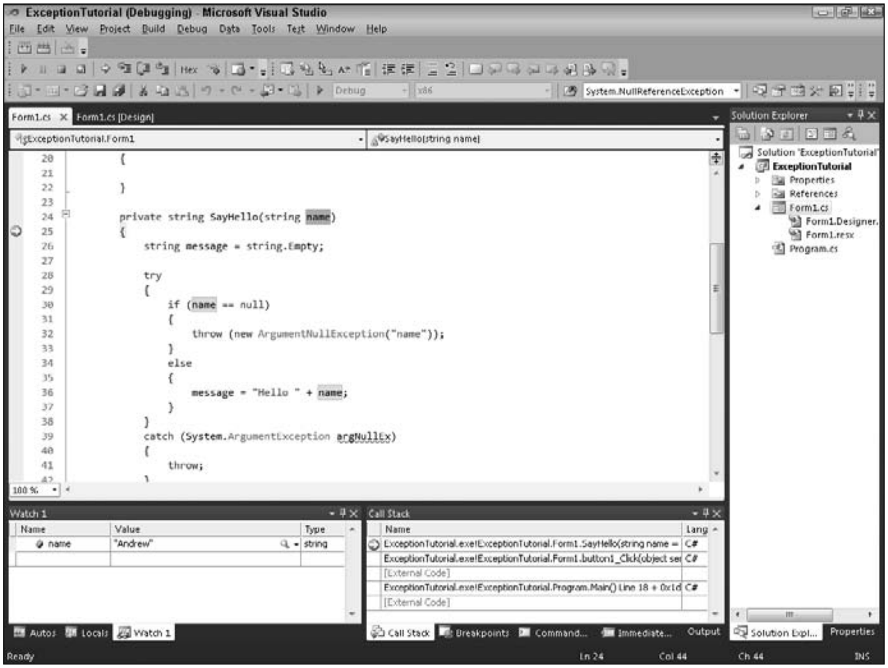
   The Debugger must be in break mode to open the Watch window. Four Watch windows are available, and each window has a number from 1 to 4 appended to its Watch name.
4. **Right-click the object that you want to add to the Watch window.**
5. **Choose Expression➪Add Watch from the shortcut menu.**

   The object appears in the Watch window.

   Be sure to click the object — not a property of the object.
6. **Press F5 to resume execution.**

   The code executes until it hits a breakpoint.

   When the variable or object is *in scope* (part of the currently executing code), its values appear in the Watch window, as you can see in Figure 7-13.



```
        {

        }

        private string SayHello(string name)
        {
            string message = string.Empty;

            try
            {
                if (name == null)
                {
                    throw (new ArgumentNullException("name"));
                }
                else
                {
                    message = "Hello " + name;
                }
            }
            catch (System.ArgumentException argNullEx)
            {
                throw;
            }
```

Name	Value	Type
name	"Andrew"	string

Name	Lang
ExceptionTutorial.exe!ExceptionTutorial.Form1.SayHello(string name =	C#
ExceptionTutorial.exe!ExceptionTutorial.Form1.button1_Click(object ser	C#
[External Code]	
ExceptionTutorial.exe!ExceptionTutorial.Program.Main() Line 18 + 0x1d	C#
[External Code]	

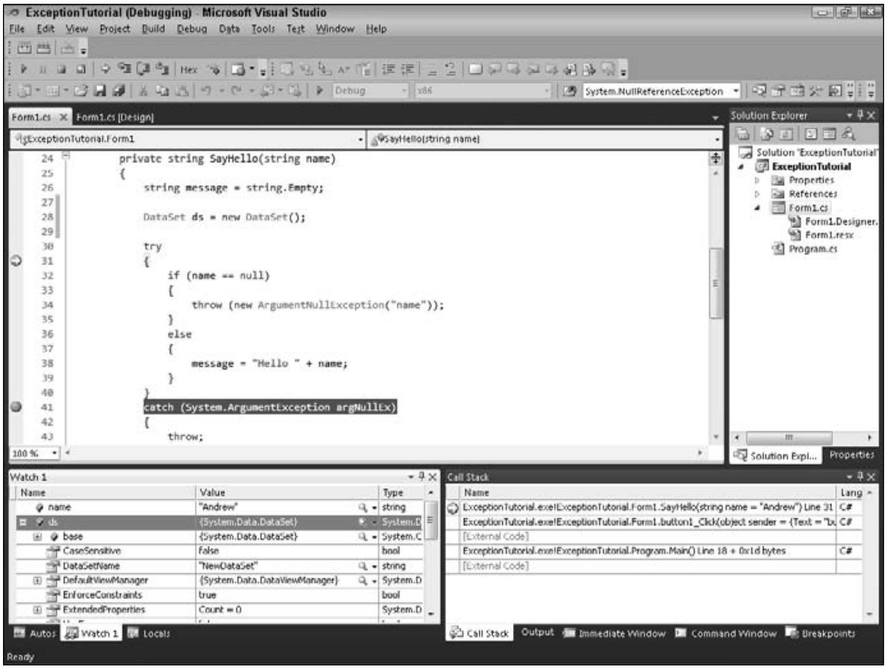
Figure 7-13: Use the Watch window to view the values of a variable or object.

You can add objects to the Watch window by typing their names in the Name field. For example, to add a Watch using a `DataSet` with the name `ds`

1. **Open the Watch window (using the preceding steps).**
2. **Type** ds **in the Name field.**

   IntelliSense lists the properties and methods that you can select.
3. **Select a member of the `DataSet` to watch.**

   Figure 7-14 shows the Watch window with `DataSet` members added.



```
private string SayHello(string name)
{
    string message = string.Empty;

    DataSet ds = new DataSet();

    try
    {
        if (name == null)
        {
            throw (new ArgumentNullException("name"));
        }
        else
        {
            message = "Hello " + name;
        }
    }
    catch (System.ArgumentException argNullEx)
    {
        throw;
```

Name	Value	Type
name	"Andrew"	string
ds	{System.Data.DataSet}	System.D
base	{System.Data.DataSet}	System.C
CaseSensitive	false	bool
DataSetName	"NewDataSet"	string
DefaultViewManager	{System.Data.DataViewManager}	System.D
EnforceConstraints	true	bool
ExtendedProperties	Count = 0	System.D

Name	Lang
ExceptionTutorial.exe!ExceptionTutorial.Form1.SayHello(string name = "Andrew") Line 31	C#
ExceptionTutorial.exe!ExceptionTutorial.Form1.button1_Click(object sender = {Text = "bu	C#
[External Code]	
ExceptionTutorial.exe!ExceptionTutorial.Program.Main() Line 18 + 0x1d bytes	C#
[External Code]	

Figure 7-14: Use IntelliSense to add members to the Watch window.

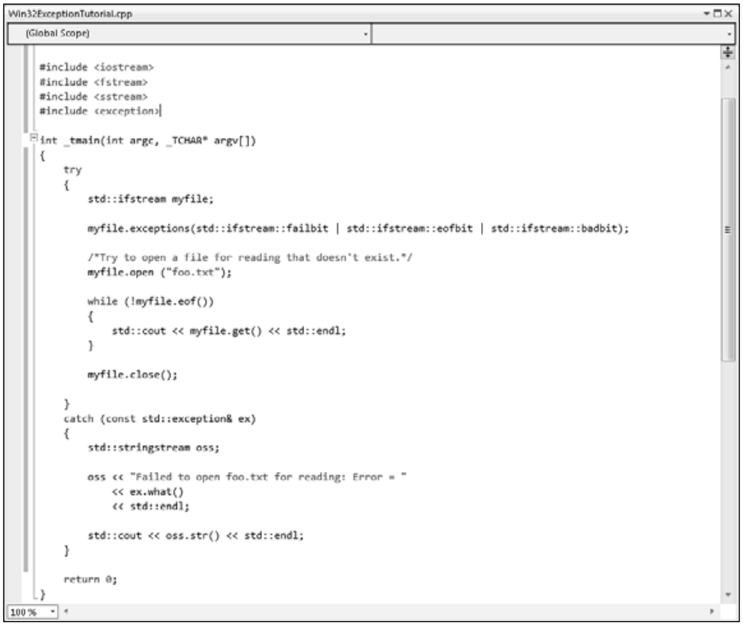
Exception handling in native C++ applications

Structure exception handling isn't limited to only applications built on the .NET framework. You can also take advantage of structured exception handling and the powerful Visual Studio debugger for your native C++ applications. Many C++ libraries that are built into Visual Studio provide structured exceptions. Some common exceptions provided by built-in libraries appear in Table 7-4.

Table 7-4 Native C++ Exception Classes

Exception Class	*Library*	*Purpose*
`std::exception`	Standard Template Library (STL)	Base class for all STL exceptions
`std::bad_alloc`	Standard Template Library (STL)	Thrown when an STL function can't allocate required memory
`CAtlException`	Active Template Library (ATL)	Base class for all ATL exceptions
`CException`	Microsoft Foundation Classes (MFC)	Base class for all MFC exceptions
`CMemoryException`	Microsoft Foundation Classes (MFC)	Out-of-memory exception
`CFileException`	Microsoft Foundation Classes (MFC)	File exceptions

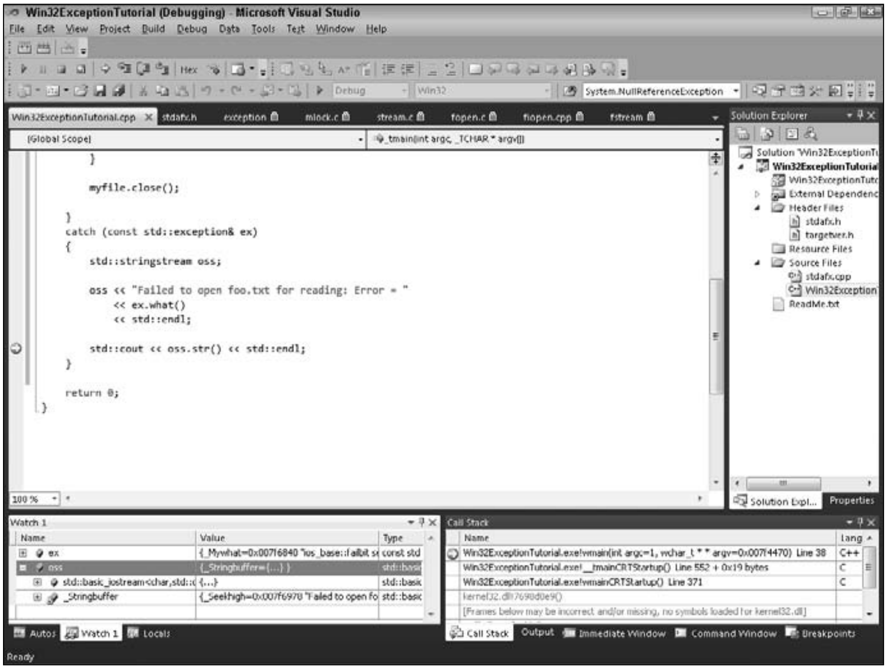
Consider the simple console application code in Figure 7-15 that attempts to open a file and output the contents to the console window.



```
#include <iostream>
#include <fstream>
#include <sstream>
#include <exception>

int _tmain(int argc, _TCHAR* argv[])
{
    try
    {
        std::ifstream myfile;

        myfile.exceptions(std::ifstream::failbit | std::ifstream::eofbit | std::ifstream::badbit);

        /*Try to open a file for reading that doesn't exist.*/
        myfile.open ("foo.txt");

        while (!myfile.eof())
        {
            std::cout << myfile.get() << std::endl;
        }

        myfile.close();

    }
    catch (const std::exception& ex)
    {
        std::stringstream oss;

        oss << "Failed to open foo.txt for reading: Error = "
            << ex.what()
            << std::endl;

        std::cout << oss.str() << std::endl;
    }

    return 0;
}
```

Figure 7-15: Native C++ application with structured exception handling.

In the code in Figure 7-15, the file `foo.txt` doesn't exist, so the code flows into the structured exception handler in the `catch()` block. As you can see, the Visual Studio debugger provides you the same features as it does for debugging .NET applications. Figure 7-16 shows an example of debugging a native C++ application.



```
        }

        myfile.close();

    }
    catch (const std::exception& ex)
    {
        std::stringstream oss;

        oss << "Failed to open foo.txt for reading: Error = "
            << ex.what()
            << std::endl;

        std::cout << oss.str() << std::endl;
    }

    return 0;
}
```

Watch 1

Name	Value	Type
ex	{_Mywhat=0x00716840 "ios_base::failbit s	const std
oss	{_Stringbuffer={...} }	std::basi
std::basic_iostream<char,std::c	{...}	std::basic
_Stringbuffer	{_Seekhigh=0x007f6978 "Failed to open fo	std::basic

Call Stack

Name	Lang
Win32ExceptionTutorial.exe!wmain(int argc=1, wchar_t * * argv=0x007f4470) Line 38	C++
Win32ExceptionTutorial.exe!__tmainCRTStartup() Line 552 + 0x19 bytes	C
Win32ExceptionTutorial.exe!wmainCRTStartup() Line 371	C
kernel32.dll!7698d0e9()	
[Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]	

Figure 7-16: Debugging native C++ applications.

Debugging for those special circumstances

The Visual Studio Debugger is sophisticated, and you can use it to debug very complex scenarios. Using the Debugger in a single-tier Windows application is about as easy as it gets. In the real world, however, applications aren't quite that simple.

You can use the Visual Studio Debugger to debug any of these items:

- ASP.NET Web sites
- Managed stored procedures
- Scripts
- Code on remote servers

Going through the setup processes for each of these is beyond the scope of this book. See the Visual Studio documentation for more information.

After you get the Debugger set up to work in your scenario, controlling execution and watching variables are basically the same.

Debugging generated code

Visual Studio generates a lot of code for you. Sometimes, you'll want to step through that code with the Debugger so you can see how the code works. To step into code that is not yours, however, you must disable the Debugger's Just My Code feature.

To disable the feature, follow these steps:

1. **Choose Tools➪Options.**

   The Options window appears.
2. **Click Debugging.**
3. **Clear the Enable Just My Code check box.**
4. **Click OK.**

Now, you can hit breakpoints inside generated code. To see generated code, click the Show All Files button on the Solution Explorer toolbar.

Chapter 8: Testing Code with Visual Studio

In This Chapter

- **Understanding different kinds of software testing**
- **Creating and running unit tests**
- **Introducing stubs and mocks**
- **Using the Object Test Bench**
- **Exploring unit-testing frameworks**

So you're plugging along coding like crazy. You'll have this set of requirements knocked out in no time. But how do you know when you're done? If you're unit testing your code while you go, the answer is when your code passes all your tests. If you're not unit testing while you go, you'll probably code until you feel like you're done and then pass the code off for somebody else to test.

Testing is an important part of the software development life cycle. Many kinds of tests are performed to ensure an application fulfills its requirements. Developers are usually responsible for writing unit tests to test the code they write. In Visual Studio 2010 Professional Edition, testing is more of a coding practice than a feature. Microsoft provides upgraded editions of Visual Studio that provide extensive support for all kinds of testing.

This chapter discusses the importance of testing, shows you how to write simple unit tests, and introduces you to some of the Visual Studio tools you can use to support your testing efforts.

Defining Unit Testing

Testing means different things, depending on the context in which it's used. Developers are usually responsible for testing the code they write — *unit testing.* After code passes unit tests, it's usually checked into a source code control repository. At some point, the entire system under development is compiled, and quality assurance testers perform even more testing.

The kinds of tests you might encounter include

- **Unit tests:** These programs are written by developers to test the code they write.
- **Integration tests:** These test units of code after they're integrated with each other.
- **System tests:** During *system tests,* the entire integrated system is tested. Depending on the kind of software being tested, system tests might include user interface testing, regression testing, and load testing.
- **Acceptance tests:** During *acceptance tests,* any or all of the system's stakeholders might participate in testing the software in a lab.

When software progresses through testing, tests become less focused on the inner workings of the code. As a result, testing becomes less automated and requires more user interaction. In the case of acceptance testing, a test lab is often set up where end users come in and bang away at the system for weeks at a time, uncovering bugs and functional shortcomings.

As a developer, you're actively involved in writing and running unit tests. Your interaction with the testing process beyond unit testing depends on several factors, including the extent to which your code provides any of the system's core functionality.

Of course, use of unit testing doesn't mean that no bugs exist. Rather, the implicit understanding is that when you "check in" your code, you're telling other developers that the code works at some basic level of functionality. Code that breaks every time that it's called by another developer shows a pretty good indication that you're either not unit testing at all or not doing enough unit testing.

If you've worked as a tester, you can likely spot those developers who never run unit tests. Not only does their code usually not conform to the requirements, but it usually blows up as soon you try to test it. Other developers, however, are very conscientious about unit testing their code. As a result, you won't need to interact with those developers as much. In other words, if you don't want quality assurance and your fellow developers on your back, unit test your code.

By definition, unit testing is about writing code. When you write a unit test, you're writing code to test code. You write a unit test to test a single unit of your code. For example, assume that your project includes code for a `Customer` object and an `Address` object. You should have one unit test for each object. The unit test contains multiple tests to test the various methods and properties of your object. Figure 8-1 shows an example.

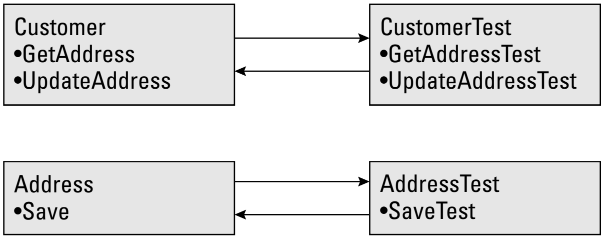


Figure 8-1: Write at least one unit test for each object in your code.

You'll find tons of rules about how you should write your tests. However, they're not really rules: They're more like opinions. And everybody has an opinion on what it means to unit test. Some good guidelines to follow are

- All tests should be automated and run as part of a suite.
- All your code should be tested.
- You should write your tests before you write your code.

REMEMBER

Getting started with unit testing can be quite overwhelming. Although these approaches are valid and ones to which you should aspire, it's better to

- Write imperfect tests than to write no tests at all.
- Run tests manually before you check in your code than run no tests at all. Test the riskiest parts of your code first until you get the hang of unit testing.

Unit Testing in Visual Studio

Unit testing is an important good coding practice. While you're coding, you should write tests that make sure your code works as you expect. Unit tests aren't supposed to test every aspect of the system. Rather, unit tests are sanity checks you use to make sure that your code works. For example, if your requirement states that the function should return an integer, you might create a unit test to test that the value returned by your function is indeed an integer.

When used properly, unit tests help you achieve the following important goals:

- Write better code.
- Have a starting point for testing code.

- Keep the development process flowing.
- Increase your confidence in your code.

Lots of patterns and frameworks are available for writing unit tests. At its simplest, a unit test simply tests that your code does what it says it does. Unit tests either pass or fail. There's no in-between.

Creating unit tests

Before you can write a unit test, you need code to test. This section uses a very simple `Hello World` example with the following methods:

- `CreateMessage()`: Creates a `Hello World` message, depending on whether the user supplies a name.
- `SayHello()`: Displays a `Hello World` message in a message box.

These methods are used in a Windows Form that displays a `Hello World` message. If the user enters a name in a text box on the form, the user's name appears in the message.

Listing 8-1 shows the code to be unit tested. Notice the two methods to test.

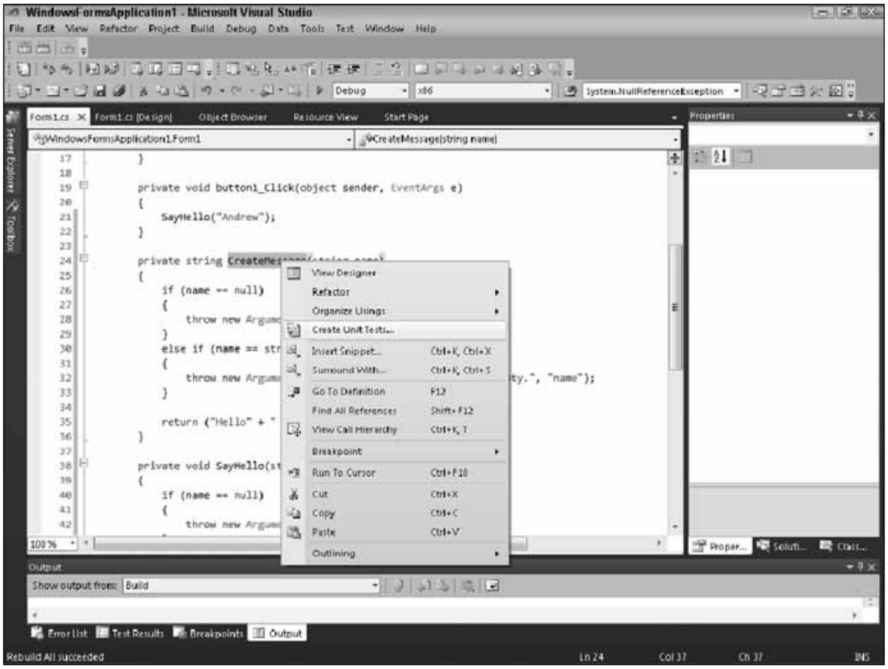
Listing 8-1: Sample Unit Test Code

```
private string CreateMessage(string name)
 {
    if (name == null)
    {
         throw new ArgumentNullException("name");
    }
    else if (name == string.Empty)
    {
         throw new ArgumentException("Parameter must not be empty.", "name");
    }

    return ("Hello" + " " + name + "!");
 }

 private void SayHello(string name)
 {
    if (name == null)
    {
        throw new ArgumentNullException("name");
    }
    else if (name == string.Empty)
    {
        throw new ArgumentException("Parameter must not be empty.", "name");
    }

    MessageBox.Show(CreateMessage(name), "Sample Application",
     MessageBoxButtons.OK, MessageBoxIcon.Information);

 }
```

To test this code, you write a set of tests that makes sure that the output is as expected. For example, the `SayHello()` method is supposed to return `Hello World`. The unit test should test the return value. Visual Studio 2010 automates this unit testing for you.

To create unit tests for these two methods:

1. **Select the methods in the code editor.**
2. **Right-click and choose Create Unit Tests from the contextual menu, shown in Figure 8-2.**



```
        }

        private void button1_Click(object sender, EventArgs e)
        {
            SayHello("Andrew");
        }

        private string CreateMe
        {
            if (name == null)
            {
                throw new Argum
            }
            else if (name == str
            {
                throw new Argum                                    ty.", "name");
            }

            return ("Hello" + "
        }

        private void SayHello(st
        {
            if (name == null)
            {
                throw new Argum
```

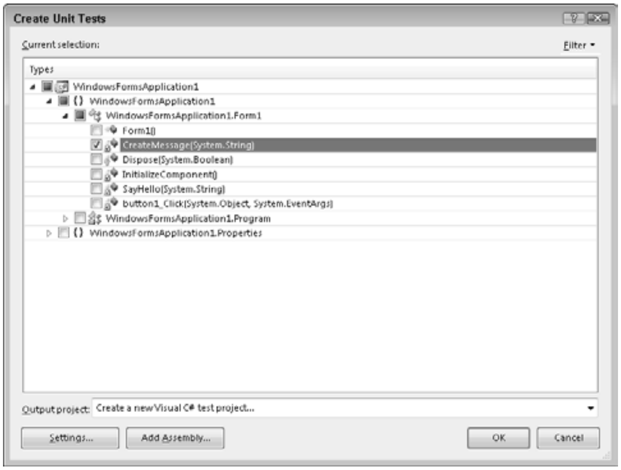
Figure 8-2: Creating unit tests from the code editor.

Visual Studio displays the Create Unit Tests dialog box, shown in Figure 8-3.

3. **Click OK.**

   Visual Studio creates a new Visual C# project in your solution that creates a test harness to unit tests for the methods you selected in Figure 8-3. Figure 8-4 shows this project in the Solution Explorer.

Each test you create is a *test case*. Some functions might require multiple test cases. Visual Studio automatically creates unit tests for you, as shown in Listing 8-2.



Create Unit Tests

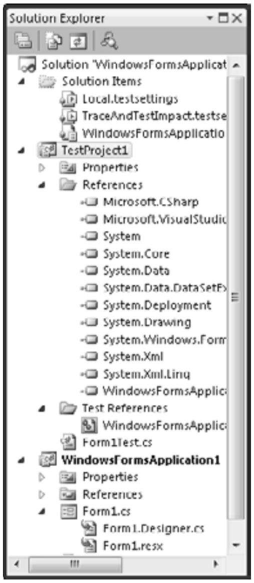
Current selection:

Types

- WindowsFormsApplication1
  - {} WindowsFormsApplication1
    - WindowsFormsApplication1.Form1
      - □ Form1()
      - ☑ CreateMessage(System.String)
      - □ Dispose(System.Boolean)
      - □ InitializeComponent()
      - □ SayHello(System.String)
      - □ button1_Click(System.Object, System.EventArgs)
    - □ WindowsFormsApplication1.Program
  - □ {} WindowsFormsApplication1.Properties

Output project: Create a new Visual C# test project...

Figure 8-3: The Create Unit Tests dialog box.



Solution Explorer

- Solution 'WindowsFormsApplicat
  - Solution Items
    - Local.testsettings
    - TraceAndTestImpact.testse
    - WindowsFormsApplicatio
  - TestProject1
    - Properties
    - References
      - Microsoft.CSharp
      - Microsoft.VisualStudic
      - System
      - System.Core
      - System.Data
      - System.Data.DataSetEx
      - System.Deployment
      - System.Drawing
      - System.Windows.Form
      - System.Xml
      - System.Xml.Linq
      - WindowsFormsApplic
    - Test References
      - WindowsFormsApplic
    - Form1Test.cs
  - **WindowsFormsApplication1**
    - Properties
    - References
    - Form1.cs
      - Form1.Designer.cs
      - Form1.resx

Figure 8-4: The unit test project in Solution Explorer.

Listing 8-2: Code to Test the Set of Code

```
namespace TestProject1
{
    /// <summary>
    ///This is a test class for Form1Test and is intended
    ///to contain all Form1Test Unit Tests
```

```
///</summary>
[TestClass()]
public class Form1Test
{
    private TestContext testContextInstance;

    /// <summary>
    ///Gets or sets the test context which provides
    ///information about and functionality for the current test run.
    ///</summary>
    public TestContext TestContext
    {
        get
        {
            return testContextInstance;
        }
        set
        {
            testContextInstance = value;
        }
    }

    #region Additional test attributes
    //
    //You can use the following additional attributes as you write your
tests:
    //
    //Use ClassInitialize to run code before running the first test in the
class
    //[ClassInitialize()]
    //public static void MyClassInitialize(TestContext testContext)
    //{
    //}
    //
    //Use ClassCleanup to run code after all tests in a class have run
    //[ClassCleanup()]
    //public static void MyClassCleanup()
    //{
    //}
    //
    //Use TestInitialize to run code before running each test
    //[TestInitialize()]
    //public void MyTestInitialize()
    //{
    //}
    //
    //Use TestCleanup to run code after each test has run
    //[TestCleanup()]
    //public void MyTestCleanup()
    //{
    //}
    //
    #endregion


    /// <summary>
    ///A test for CreateMessage
    ///</summary>
    [TestMethod()]
```

(continued)

Book III
Chapter 8

Testing Code
with Visual Studio

Listing 8-2 *(continued)*

```
        [DeploymentItem("WindowsFormsApplication1.exe")]
        public void CreateMessageTest()
        {
            Form1_Accessor target = new Form1_Accessor(); // TODO: Initialize to
    an appropriate value
            string name = string.Empty; // TODO: Initialize to an appropriate
    value
            string expected = string.Empty; // TODO: Initialize to an appropriate
    value
            string actual;
            actual = target.CreateMessage(name);
            Assert.AreEqual(expected, actual);
            Assert.Inconclusive("Verify the correctness of this test method.");
        }

        /// <summary>
        ///A test for SayHello
        ///</summary>
        [TestMethod()]
        [DeploymentItem("WindowsFormsApplication1.exe")]
        public void SayHelloTest()
        {
            Form1_Accessor target = new Form1_Accessor(); // TODO: Initialize to
    an appropriate value
            string name = string.Empty; // TODO: Initialize to an appropriate
    value
            target.SayHello(name);
            Assert.Inconclusive("A method that does not return a value cannot be
    verified.");
        }
    }
}
```

The `Assert` class in the `Microsoft.VisualStudio.TestTools.UnitTesting` namespace provides a set of static methods that provide you the conditions for your unit tests. Methods on the `Assert` class throw the `AssertFailedException` if the condition specified isn't true. Table 8-1 lists the methods on the `Assert` class.

Table 8-1 Assert Class Static Methods

Method	*Description*
`AreEqual`	Verifies that specified values are equal
`AreNotEqual`	Verifies that specified values aren't equal
`AreNotSame`	Verifies that specified object variables refer to different objects
`AreSame`	Verifies that specified objects refer to the same object

Method	Description
`Fail`	Fails an assertion without checking any conditions
`Inconclusive`	Indicates that an assertion can't be proven true or false. Also used to indicate an assertion that hasn't yet been implemented
`IsFalse`	Verifies that a specified condition is false
`IsInstanceOfType`	Verifies that a specified object is an instance of a specified type
`IsNotInstanceOfType`	Verifies that a specified object isn't an instance of a specified type
`IsNotNull`	Verifies that a specified object isn't null
`IsNull`	Verifies that a specified object is null
`IsTrue`	Verifies that a specified condition is true

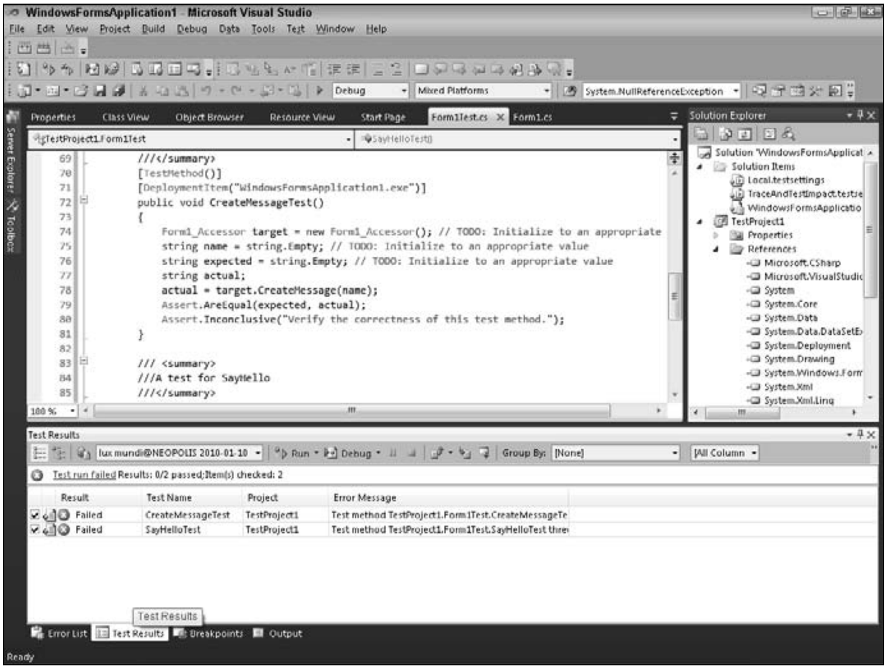
Running a battery of tests

You should be able to run your tests with the click of a mouse or by entering a single command. To run these tests, right-click the test project in Solution Explorer and choose Debug➪Start New Instance. The unit test project executes and displays the results in the Test Results menu, as shown in Figure 8-5.

Based on the results in Figure 8-5, it looks like the unit tests aren't complete enough to adequately test the code. In this case, both unit tests failed because `SayHello()` and `CreateMessage()` throw `System.ArgumentException` if the parameter `"name"` is empty. In this case, you have to tell the unit test framework that you're expecting these methods to throw exceptions if the parameter `"name"` is null or empty. To do so, you add an `ExpectedException` attribute to each of the test methods:

```
[ExpectedException(typeof(System.ArgumentException))]
      public void SayHelloTest()…
```

Adding an `ExpectedException` tells the unit test framework that you're expecting your methods to throw an exception of type `System.Argument Exception` or `System.ArgumentNullException` because it's derived from `System.ArgumentException`. Now, when you pass `string.Empty` or null to these methods, the unit tests won't fail because you're expecting these methods to throw exceptions when these values are provided for the parameter `"name"`. So to adequately test these methods, you need to provide an empty string, a null string, and a valid string to each of these methods. You also want to test the negative case where the expected value isn't equal to the actual value. Listing 8-3 shows the complete unit test for `CreateMessage`.



```
        ///</summary>
        [TestMethod()]
        [DeploymentItem("WindowsFormsApplication1.exe")]
        public void CreateMessageTest()
        {
            Form1_Accessor target = new Form1_Accessor(); // TODO: Initialize to an appropriate
            string name = string.Empty; // TODO: Initialize to an appropriate value
            string expected = string.Empty; // TODO: Initialize to an appropriate value
            string actual;
            actual = target.CreateMessage(name);
            Assert.AreEqual(expected, actual);
            Assert.Inconclusive("Verify the correctness of this test method.");
        }

        /// <summary>
        ///A test for SayHello
        ///</summary>
```

Test run failed Results: 0/2 passed;Item(s) checked: 2

	Result	Test Name	Project	Error Message
☑	Failed	CreateMessageTest	TestProject1	Test method TestProject1.Form1Test.CreateMessageTe
☑	Failed	SayHelloTest	TestProject1	Test method TestProject1.Form1Test.SayHelloTest thre

Figure 8-5: The Test Results Window in Visual Studio.

Listing 8-3

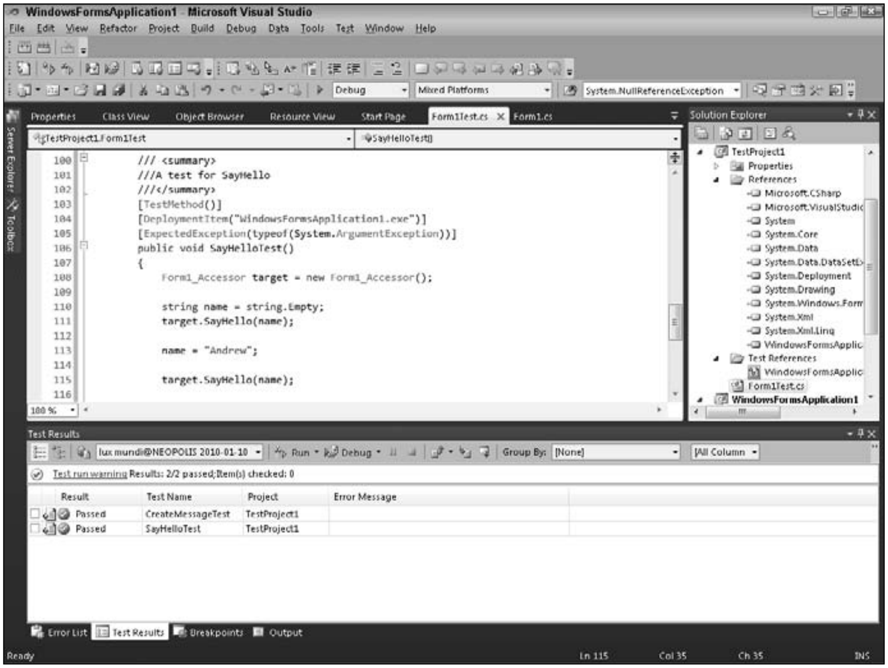
```
/// <summary>
        ///A test for CreateMessage
        ///</summary>
        [TestMethod()]
        [DeploymentItem("WindowsFormsApplication1.exe")]
        [ExpectedException(typeof(System.ArgumentException))]
        public void CreateMessageTest()
        {
            Form1_Accessor target = new Form1_Accessor();
            string name = "Andrew";
            string expected = "Hello Andrew!";
            string actual;
            actual = target.CreateMessage(name);
            Assert.AreEqual(expected, actual);

            name = string.Empty;
            expected = string.Empty;

            actual = target.CreateMessage(name);
            Assert.AreEqual(expected, actual);

            name = null;
            expected = null;
            actual = target.CreateMessage(name);
```

Book III
Chapter 8

Testing Code with Visual Studio

```
        Assert.AreEqual(expected, actual);

        name = "Andrew";
        expected = "Hello Joe!";

        Assert.AreNotEqual(expected, actual);
    }
```

The first test sets the parameter `"name"` to `"Andrew"`. In this test, you expect the resulting message to be `"Hello Andrew!"` as indicated by the `Assert.AreEqual()`. The second test passes an empty string as the `"name"` parameter. Because you provided the `ExpectedException` attribute for the `System.ArgumentException`, this tests two passes. The third test passes `null` as the `"name"` parameter. This time, too, the results are what you expect because `CreateMessage()` throws a `System.ArgumentNullException`. The final test tests the negative case where `"Andrew"` is again passed to `CreateMessage()`, and you test to ensure that the result isn't equal to something other than `"Hello Andrew!"`. In this case, `"Hello Joe!"` doesn't match, and the `Assert.AreNotEqual` is true. Now, in Figure 8-6, Visual Studio shows that your tests have succeeded.



```
        /// <summary>
        ///A test for SayHello
        ///</summary>
        [TestMethod()]
        [DeploymentItem("WindowsFormsApplication1.exe")]
        [ExpectedException(typeof(System.ArgumentException))]
        public void SayHelloTest()
        {
            Form1_Accessor target = new Form1_Accessor();

            string name = string.Empty;
            target.SayHello(name);

            name = "Andrew";

            target.SayHello(name);

```

Test run warning Results: 2/2 passed;Item(s) checked: 0

Result	Test Name	Project	Error Message
Passed	CreateMessageTest	TestProject1	
Passed	SayHelloTest	TestProject1	

Figure 8-6: Display the results of your successful unit tests.

Approaches to Unit Testing

Within the past few years, unit testing has taken on a life of its own. You're not cool if you're not doing a unit test. One of the hardest choices when creating unit tests is deciding what to test. When you examine code that you want to test, consider the following issues:

- **Public methods:** *Public methods* are those features of your code that are exposed to the outside world. Pretty much everyone agrees you should unit test your public methods. After all, a public method is sitting there saying, "Call me."
- **Private and protected methods:** *Private and protected methods* in your code do the work of your public methods. As a result, some people believe that testing private and protected methods is wasteful. They believe that unit testing public methods implicitly tests private and protected methods. Plus, testing private and protected methods is harder. By their nature, private and protected methods are, well, private and protected. Unlike public methods, they aren't saying, "Call me." Just the opposite is true, so you need to go out of your way to test them.
- **Interactions with other code:** Unless you're writing all your application's features in a single function, chances are high your code needs to interact with other code. The very nature of unit testing, however, is that you want to test only a single unit of code. How can you do that if you have to interact with other code?

Public methods are easily tested because, by definition, you can call a public method from anywhere in your code.

People take all kinds of approaches to solve the problems of testing private methods or code that interacts with other code. One popular approach is to use software that specializes in unit testing, usually called *unit testing frameworks.* (See the section "Automating Tests with Testing Frameworks," later in this chapter.)

Two approaches to testing code that interacts with other code are *stubs* and *mocks.*

Letting stubs do the tough testing

By definition, a unit test is supposed to test only one unit of code. No unit of code lives in isolation, however. Most developers work around this pesky paradox by using *code stubs,* which act as placeholders for units of code that the unit being tested needs to interact with.

The code stub placeholder is better than interacting with the actual units of code because each unit of code introduces a new set of variables. Instead, you can program the code stub to return a consistent set of values against which the unit can interact. Returning a consistent set of values creates predictability, which makes it easier to create tests that you can repeat.

In reality, developers who use stubs usually go ahead and test the real units of code that interact with the unit they're testing — that is, as long as it's simple to do so. For example, say that you create a Customer object with a public method called GetAddress. GetAddress returns an Address object. The Customer and Address objects are each units of code that have their own set of unit tests. Many developers will create a test that allows their Customer object to interact with the Address object even though the test extends beyond the unit they want to test.

When the developer encounters a set of code that is too complicated or messy to interact with a simple unit test, the developer usually creates a stub. The stub returns a set of canned values that the developer expects to get from the code.

For example, say that your Customer object has a method called UpdateAddress. UpdateAddress accepts a new address for a customer and passes that data off to an Address object. The Address object calls a service that validates the address against a valid ZIP code database. You don't want to have to deal with all that when you're testing your Customer object.

Instead of using the Address object, you create a stub that returns two possible values. If you pass a valid address, it returns the value ValidAddress. If you pass an invalid address, it returns the value InvalidAddress.
So, how does the stub know what's a valid address and what's an invalid address? You specify in the stub that 123 Main Street, for example, is a valid address. Everything else is invalid. Figure 8-7 shows an example of a stub.

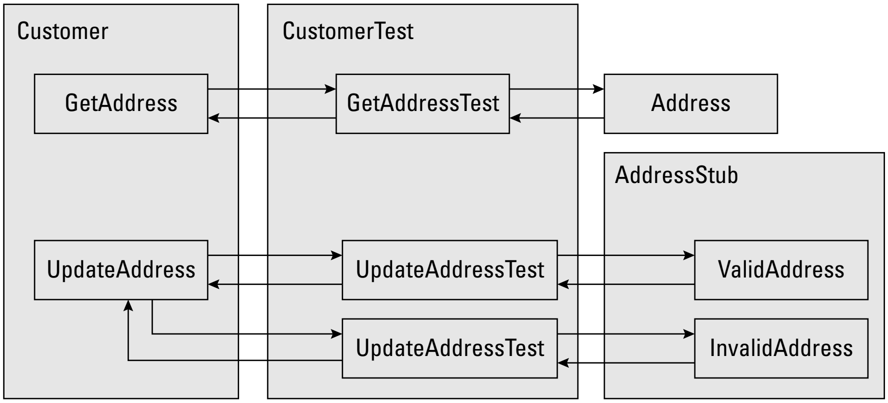


Figure 8-7: Use stubs any time you want to avoid messy code interactions in your testing.

Simplifying testing with mocking objects

Other developers believe that creating a bunch of objects and stubs is too much work. These developers use a special library to create mock objects. With mock objects — *mocks* — you tell the object which methods you expect it to run, how many times the methods will run, and what values to return.

For example, say that you use a mock `Address` object to interact with your `Customer` address. For your `UpdateAddress` test, you tell the mock `Address` object to call the `Address` object's `Save` method. Your test tells the mock object that you're going to send a value of `123 Main Street` to the `Save` method and that you expect it to return a value of `true`. Your test also tells the mock object that you're going to send the value `123 Nowhere Boulevard` and that you expect the `Save` method to return the value `false`. Figure 8-8 shows an example of a mock object.

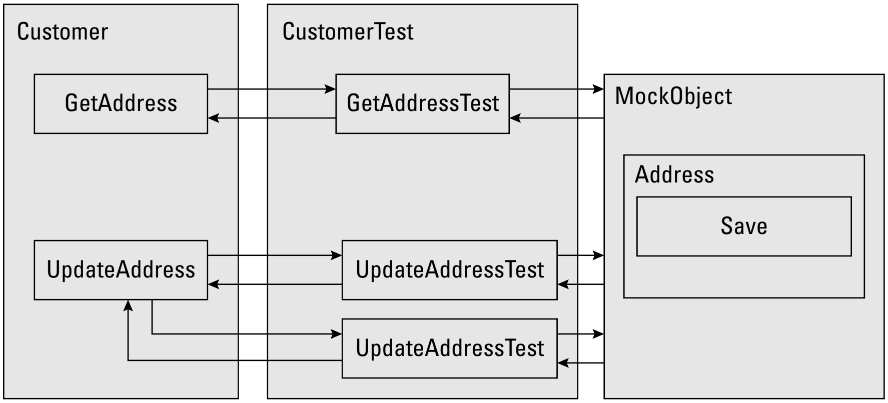


Figure 8-8: Use mock objects to set expectations about methods called and values returned.

Stubs versus mocks

Deciding which method to use can be a little difficult at first glance, but several key differences exist between using mocks and stubs, based on what you want to test for. Here are some of them:

✦ **Mocks**

- You tell your mock objects which methods to call, how many times they'll be called by your test, and what return values you expect.
- You create mocks for all objects with which your code interacts.
- Mocks require a library of code to create the mock objects.

- **Stubs**
    - Stubs usually return only values.
    - You create stubs only when necessary.

People who use stubs tend to test clusters of interacting objects together. For example, they might test all the code related to customers and orders together because those units of code interact.

People who use mocks, on the other hand, are more likely to take a top-down approach. They might start testing at the user interface level first and then use mock objects to mimic the behavior of business rules or data access code. They are likely to test first and code second. In other words, they use testing as a way to discover what code they still need to write to make their system work.

Using mock objects is very popular in the test-driven development (TDD) style. With TDD, you test first and code second. For more information, see the Test Driven Web site at `www.testdriven.com`.

Automating Tests with Testing Frameworks

Unit tests are often grouped so that they can be executed by a testing framework. A framework isn't necessary, but using a framework allows you to do these things:

- Formalize unit testing.
- Create consistency among unit tests.
- Make automating testing easier.
- Execute groups of tests at one time.

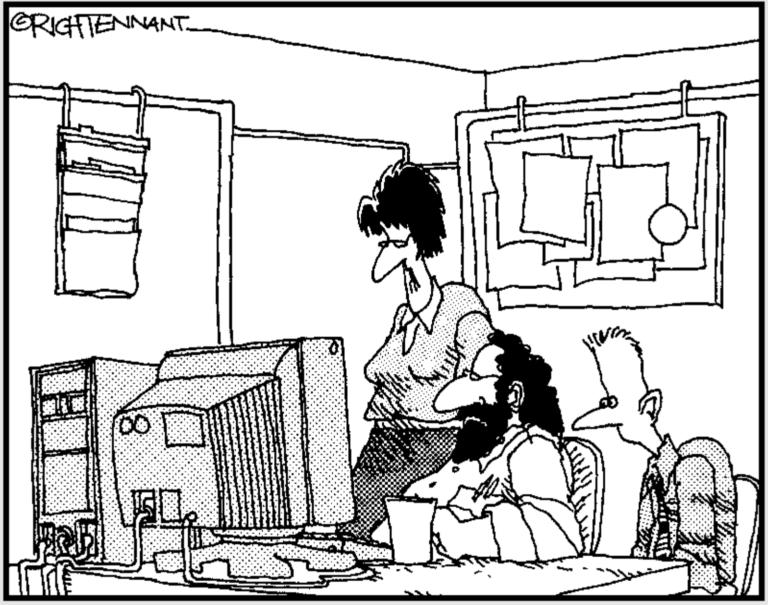
Frameworks are a necessity for most systems. Because unit tests are so small in nature, even a simple system has lots of tests. In most cases, using a framework to administer the tests makes sense. Unit testing frameworks exist for nearly every programming language under the sun. Popular unit testing frameworks for the .NET Framework include

- **NUnit:** By far, NUnit is the most-popular testing framework around. NUnit is an open source (free) framework. However, a drawback to NUnit is that it isn't integrated into Visual Studio. Still, many folks use NUnit to manage their unit tests. You can visit the NUnit Web site at `www.nunit.org`.

- **NMock:** Creates mock objects for use with unit tests.
- **TestDriven.NET:** Integrates testing frameworks, such as NUnit, into the Visual Studio development environment. TestDriven.NET gives you the ability to access testing commands from a shortcut menu in the code editor. You can visit the TestDrive.NET Web site at `www.testdriven.net`.

Book IV

Basics of Building Applications with Visual Studio 2010



The 5th Wave

By Rich Tennant

"What I'm looking for are dynamic software applications and content, not software innuendoes and intent."

Contents at a Glance

Chapter 1: Getting Smart about Smart Clients

In This Chapter

- **Getting acquainted with Windows Forms**
- **Creating your first smart client application**
- **Working with controls**
- **Setting properties and responding to events**

Visual Studio has all the tools you need to build everything from the simplest to the most complex Windows applications. By using the Microsoft .NET Framework, Visual Studio provides the visual designers that make building Windows applications a breeze. For example, Visual Studio gives you the following elements to accomplish common tasks:

- Project Templates help you set up your project.
- Visual designers help you lay out Windows Forms.
- The control toolbox makes adding user interface (UI) elements to your Windows Forms a breeze.
- Wizards, property-setting windows, and shortcut task lists enable you to easily configure UI elements.

This chapter walks you through some of the tools that Visual Studio provides for building managed Windows applications that use either Visual Basic or C#.

Switching from Fat Clients to Smart Clients

The Windows operating system (OS) and applications that run on Windows have been around for more than 30 years. With ubiquity, early Windows applications became bloated and hard to deploy. In response, developers snubbed their noses at these newly dubbed fat clients and started using a combination of client and server resources to create leaner applications.

Proving that the grass isn't always greener, folks quickly learned that thin clients had their problems, too. Although thin clients were easier to deploy, they weren't all that feature rich, and they didn't work when the network went down. Then came the Internet, and the world went crazy for Web-based applications. Many talented developers found innovative ways to twist and contort simple Web pages into what seemed like full-blown desktop applications. They used a combination of server-side code and JavaScript. Although developers had modest success in this arena, many diehard Windows programmers cringed at the lack of UI sophistication in Web-based applications.

Thanks to inexpensive, fast hardware and the maturity of Windows application development, the Windows application is again being redefined — this time as the smart client. Although a smart client may look like any old Windows application, you can use its key differences to

- Create rich user experiences.
- Deploy and maintain applications with greater ease.
- Enjoy more security than in a traditional Windows application.

Smart clients still provide all the same features as earlier generations of Windows applications:

- You can build data-centric, interactive applications.
- You can access the local computer.
- They don't require a server in order to run.

Windows applications are all about creating an intuitive, visually appealing, and interactive user experience. If user experience weren't important, everyone would still be running good old-fashioned, character-based applications in DOS.

Smart clients aren't the last word in Windows applications. The next wave of Microsoft products for Windows and Visual Studio will usher in even more opportunities for Windows developers to enhance the user experience.

Designing Windows Forms

The basic building block of a smart client Windows application is the Windows Form. Windows Forms, a key technology of the .NET Framework, provide

- Libraries for creating a UI
- Libraries for common tasks, such as reading and writing to the file system
- A set of controls that provide common UI elements

The Windows Forms Designer is the Visual Studio visual design surface for working with Windows Forms. Use the Windows Forms Designer for these tasks:

- Visually lay out your UI.
- Configure UI attributes.
- Write custom code that's executed when users interact with your form.

Figure 1-1 displays the Windows Forms Designer in Visual Studio. The next section walks you through creating a new Windows application and using the forms designer.

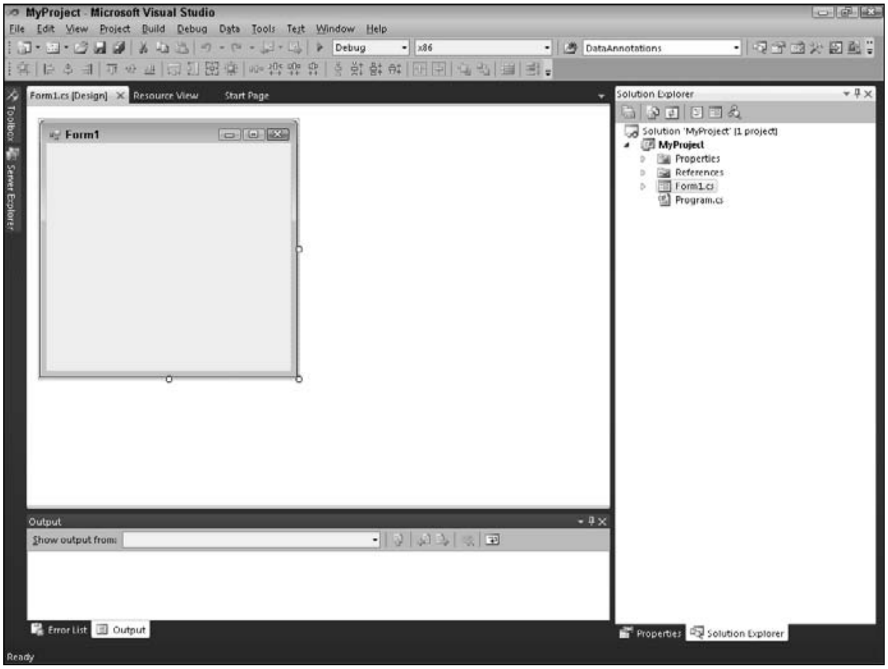


Figure 1-1: Use the Windows Forms Designer to lay out your user interface.

Creating your first smart client project

Visual Studio uses Solutions and Projects containers to organize all the files necessary to build an application. When you create a new Windows application, Visual Studio creates all the files you need and prepares the development environment with the windows, designers, and editors that you need to work with your new project.

To create a new smart client project in Visual Studio, follow these steps:

1. **Choose File⇨New⇨ Project.**

   The New Project window appears.

   You can also open the New Project window by pressing Ctrl+Shift+N. On some editions of the product, you can choose File⇨New Project to create a project.

2. **In the Installed Templates hierarchy, click the programming language you want to use.**

   A list of project templates that are available for that programming language appears on the right.

   Windows applications created by using Visual Basic, C#, managed C++, and F# use the .NET Framework. Although most of the examples in this chapter use C# and Visual Basic, you can create Windows applications by using any language. See Chapter 3 in Book III for more details about creating Windows applications with C++.

3. **Click the Windows Forms Application icon.**

4. **Enter a unique name for your application in the Name text box.**

   Although you can give your application any name you want, you probably should choose a name that describes your application. Many developers include the word Windows or the abbreviation Win in an application's name to distinguish it from other kinds of applications, such as Web or mobile applications. In this example, I use the default name `WindowsApplication1`.

5. **Click OK to create the application.**

   Visual Studio creates a new solution with a Windows project. A new Windows Form is added to the project, and Visual Studio displays the form in the Windows Forms Designer.

Saying, "Hello world!"

When you create a new Windows application, Visual Studio generates all the code necessary to display the Windows Form on the screen when you run your application. To run your new Windows application, press Ctrl+F5. Visual Studio builds your Windows application and displays a blank Windows Form.

Most Windows applications use menus, text boxes, and buttons. These types of user interface elements are controls. Follow these steps to add controls to make your application say, "Hello world!":

1. **Open the control toolbox by pressing Ctrl+Alt+X.**
2. **Make sure that you open the tab named All Windows Forms.**
3. **Drag and drop a label control from the toolbox onto the Form Designer.**
4. **Drag and drop a button control onto the forms designer.**
5. **Double-click the button control.**

   Visual Studio creates a block of code to handle the button's Click event. The Code Editor appears with the cursor flashing in the block of code.
6. **Type this line in the Code Editor:**
   - **VB**

   label1.Text = "Hello World!"
   - **C#**

   label1.Text = "Hello World!";
7. **Press Ctrl+F5 to run your Windows Form.**

When you click the button on your form, Hello World! appears on the label you add in Step 3, as shown in Figure 1-2.

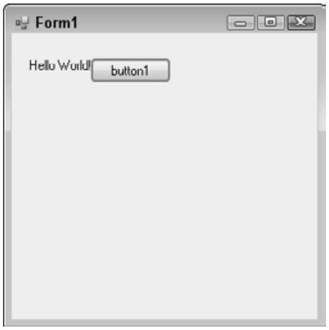


Figure 1-2: "Hello World!" appears on the form's label.

Taking Command from Control Central

A Windows Form is a blank canvas: It doesn't do much on its own. To bring your Windows Form to life, you must add controls to it. Controls are UI elements, such as buttons and text boxes, designed for interacting with your Windows Form. Controls breathe life into your Windows Forms by

- Providing structure and navigation in the form of menus, toolbars, and status bars
- Allowing users to interact with your forms by using buttons, labels, and text boxes
- Displaying dialog boxes to provide user feedback about your application's state

TIP

Putting controls on a form is also a way of designing an application. The form with controls and no code is an adequate prototype with which you can then proceed to plan the code. By taking the time to think through the application and place the controls as they will be in the final application, you'll likely have fewer bugs, flaws, and debugging issues during the development process.

Introducing the control toolbox

Visual Studio provides a toolbox chock full of controls. The control toolbox, shown in Figure 1-3, includes all the commonly used controls that even casual Windows users are familiar with, such as

- Labels, text boxes, and buttons
- Check boxes, list boxes, and combo boxes
- Calendars and date pickers

The toolbox displays controls in groups, or tabs:

- **All Windows Forms:** Contains an alphabetical list of controls.
- **Common Controls:** Lists frequently used controls, such as labels and text boxes.
- **Containers:** Used for laying out and grouping other controls on the form.
- **Menus & Toolbars:** Creates the menus and toolbars your application uses.
- **Data:** Contains controls and wizards for accessing data.

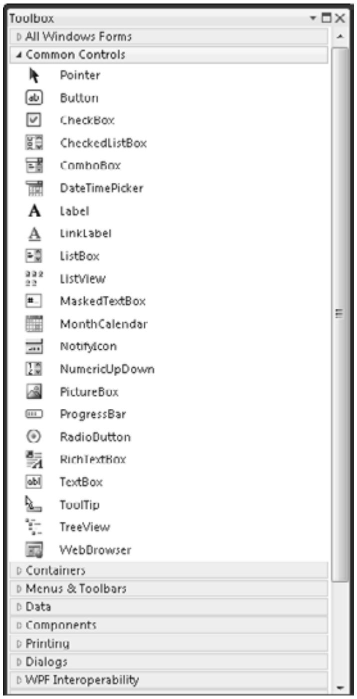


Figure 1-3: Add controls to your form from the toolbox.

- **Components:** Contains controls that provide back-end services, such as connecting to an event log.
- **Printing:** Provides printing features.
- **Dialogs:** Allows you to add common dialog boxes, such as those used for saving files.
- **WPF Interoperability:** Provides interoperability between Windows Presentation Foundation and Windows Forms.
- **Crystal Reports:** Allows you to add a Crystal Report to your form. See Book VI, Chapter 4 to read more about Crystal Reports.
- **General:** Contains an empty group.

This chapter demonstrates how to use frequently used controls, such as labels, text boxes, and buttons. More advanced controls, such as those in the containers and on the menus and toolbox tabs, are discussed in Chapter 2 of this book. To read more about using data controls on Windows Forms, see Book V, Chapter 2.

Customizing the toolbox

You can customize the toolbox by

- Adding and removing tab groups
- Moving tab groups up or down
- Adding and removing controls to and from the tab groups

To customize the toolbox, follow these steps:

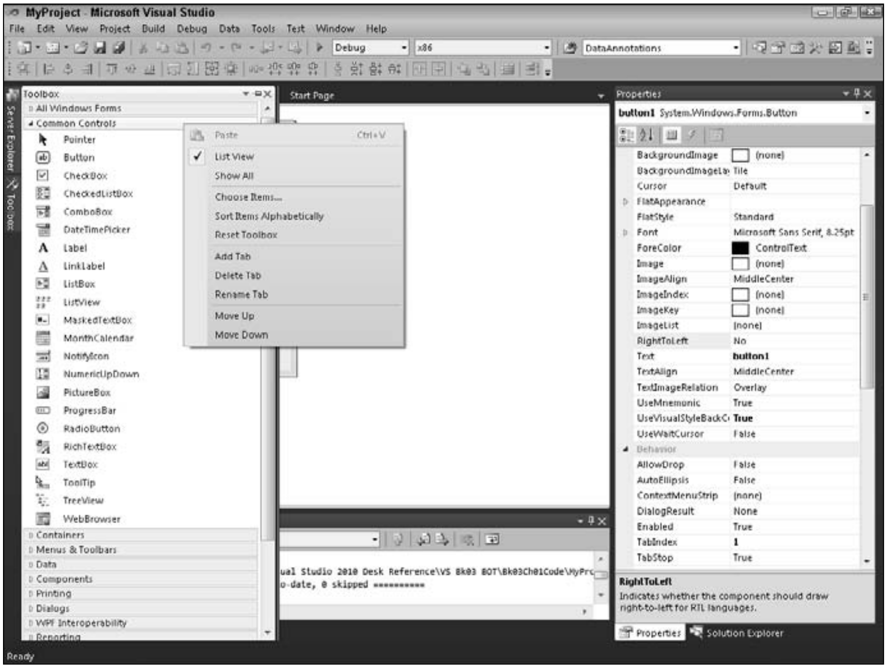
1. **Right-click an item, such as a specific control or tab, in the toolbox you want to customize.**

   A contextual menu appears, as shown in Figure 1-4.

2. **From the shortcut menu, choose the action you want to take.**

   TIP

   To add new controls to the toolbox, choose Choose Items from the contextual menu.



```
ual Studio 2010 Desk Reference\VS Bk03 BOT\Bk03Ch01Code\MyPrc
o-date, 0 skipped ==========
```

BackgroundImage	□ (none)
BackgroundImageLa	Tile
Cursor	Default
FlatAppearance	
FlatStyle	Standard
Font	Microsoft Sans Serif, 8.25pt
ForeColor	■ ControlText
Image	□ (none)
ImageAlign	MiddleCenter
ImageIndex	□ (none)
ImageKey	□ (none)
ImageList	(none)
RightToLeft	No
Text	**button1**
TextAlign	MiddleCenter
TextImageRelation	Overlay
UseMnemonic	True
UseVisualStyleBackC	**True**
UseWaitCursor	False
AllowDrop	False
AutoEllipsis	False
ContextMenuStrip	(none)
DialogResult	None
Enabled	True
TabIndex	**1**
TabStop	True

RightToLeft
Indicates whether the component should draw right-to-left for RTL languages.

Figure 1-4: Right-click a tab header to customize the tab.

Using more controls

Whenever possible, you should use the controls provided in Visual Studio. Using them creates a consistent user experience because Windows users are generally familiar with most Windows controls. Sometimes, however, you may need a control that isn't available in the Visual Studio toolbox. You have several options:

- **Build your own.** Because .NET is an object-oriented language, you can extend an existing control or build your own.
- **Use ActiveX controls.** Windows Forms can host ActiveX controls, which are built by using the Component Object Model (COM) technology.

  You add controls to the Visual Studio toolbox by choosing the Choose Toolbox Items command from the Tools menu.
- **Buy the control.** Many third-party vendors sell controls. You could also sell a control that you build.
- **Download the control.** Sometimes, you can find controls available for free or as shareware on the Internet. Be sure to read the license before you redistribute controls you find there.

You can search for controls online by using the Community Search feature. See Book II, Chapter 3 for more information.

Adding controls to your form

The easiest way to add a control to a form is to drag and drop it from the toolbox onto the Windows Form. You can also draw a control to specify its exact size. To draw a text box on a Windows Form, follow these steps:

1. **Click the text box control in the toolbox.**

   If the toolbox is closed, press Ctrl+Alt+X to display it.

2. **Move your mouse pointer to the Windows Form.**

   Note: You don't need to drag the text box.

   The pointer changes to a plus sign (+).

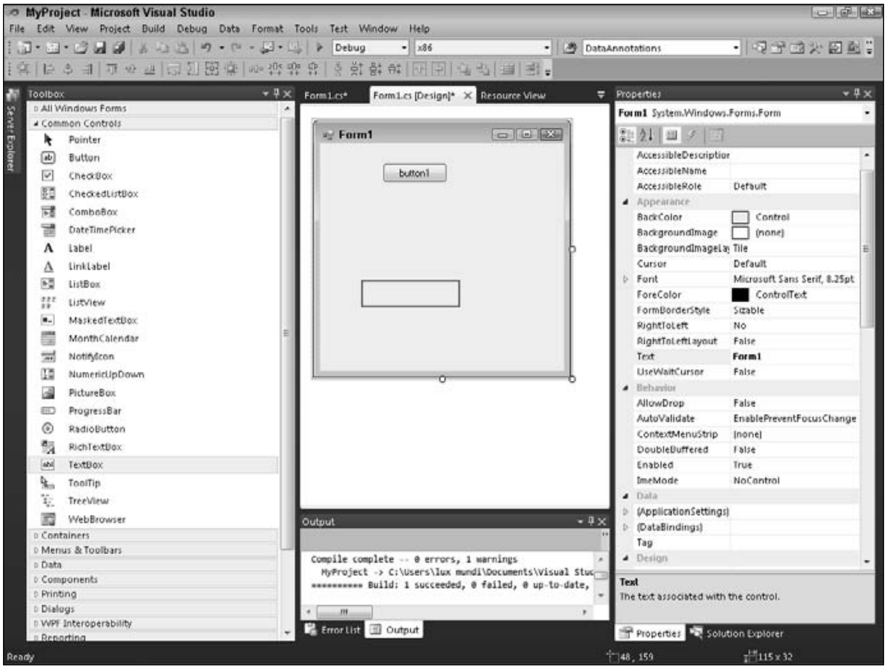
   If you change your mind about drawing the control, click a different control or click the pointer in the toolbox to reset your mouse pointer without adding a control to the form.

3. **In the upper-left corner, click the mouse pointer wherever you want to start drawing the text box.**

   To draw a default-size text box, click and release your mouse button without dragging.

4. **Drag the mouse pointer to the upper-left corner where you want the text box to end, as shown in Figure 1-5, and release the mouse button.**

   The text box is drawn on the form. When you're done placing controls, the next step is typically to format them and set how you want them to behave. See the next section, "Manipulating Controls," for details.



```
Compile complete -- 0 errors, 1 warnings
  MyProject -> C:\Users\lux mundi\Documents\Visual Stuc
========== Build: 1 succeeded, 0 failed, 0 up-to-date,
```

Properties

Form1 System.Windows.Forms.Form

AccessibleDescriptior	
AccessibleName	
AccessibleRole	Default
Appearance	
BackColor	Control
BackgroundImage	(none)
BackgroundImageLa	Tile
Cursor	Default
Font	Microsoft Sans Serif, 8.25pt
ForeColor	ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	**Form1**
UseWaitCursor	False
Behavior	
AllowDrop	False
AutoValidate	EnablePreventFocusChange
ContextMenuStrip	(none)
DoubleBuffered	False
Enabled	True
ImeMode	NoControl
Data	
(ApplicationSettings)	
(DataBindings)	
Tag	
Design	

Text
The text associated with the control.

Figure 1-5: Release the mouse pointer in the lower-left corner of the toolbox.

Manipulating Controls

Creating a UI with controls involves more than dragging and dropping. To get the most from controls, you need to know how to position them on your form. This section covers some common tools for putting controls exactly where you want them.

Formatting controls

Even the simplest Windows Form can have dozens of controls. Developers often want to align controls in a way that's visually appealing to users. Although you can work with each control individually — drag, drop, resize, and align — using the Visual Studio Format menu is much easier. After selecting two or more controls, use the Format menu to perform the following actions:

- **Align:** Position controls into alignment with one another.
- **Make Same Size:** Make controls the same width or height.
- **Horizontal Spacing:** Set the side-by-side spacing between controls.
- **Vertical Spacing:** Set the spacing above and below controls in a row.
- **Center in Form:** Position a control or group of controls in the center of the Windows Form.
- **Order:** Set controls to appear in front of other controls.
- **Lock Controls:** Lock the position of a single control or group of controls so that the position can't be changed.

Visual Studio displays a Layout toolbar with many of the common formatting commands. To display the Layout toolbar if it's not already open, follow these steps:

1. **Choose View⇨Toolbars.**

   A submenu displays a list of all Visual Studio toolbars, and check marks indicate all visible toolbars.

2. **Select the Layout option.**

   The Layout toolbar appears, as shown in Figure 1-6. You can use this toolbar to access common formatting commands.

Figure 1-6: The Layout toolbar.

Here's how to create a form with aligned labels and buttons for accepting user input:

1. **Draw three labels on the form and Stack the labels vertically.**

   Try not to align them.

2. **Draw three text buttons on the form — again, without aligning them.**

3. **Hold down the Shift key and then click the three buttons to select them.**

   You have three options for selecting formatting controls:

   - *Shift+click:* Format all controls to conform with the first control selected.
   - *Ctrl+click:* Format all controls to conform with the last control selected.
   - *Click and drag:* Click the form and then drag a selection rectangle around all the controls you want to select.

4. **Click the Make Same Size icon on the Layout toolbar.**

   The buttons are made the same size.

5. **Click the Align Lefts icon on the Layout toolbar.**

   Align Lefts is the second icon on the toolbar. The buttons are aligned to the left.

6. **Position the mouse pointer over the right side of the buttons. When the pointer turns to an arrow, drag your mouse to the right and left.**

   The buttons widen and narrow with the mouse movement.

7. **Click the form to deselect the buttons.**

8. **Hold down the Shift key while you select the three label controls.**

9. **Choose Format➪Align➪Rights.**

   The labels are aligned to the right.

10. **Position the pointer over the labels until it turns into a four-way arrow and drag the labels to the left to create space between the labels and buttons.**

    You can use the arrow keys on your keyboard to move the labels around on the form.

Seeing the snaplines

When you add a control to a form, Visual Studio provides visual cues, or *snaplines,* for positioning the control. Snaplines recommend the best position for the controls in accordance with spacing recommendations set in the official Microsoft UI guidelines. The snaplines appear while you drag controls around the form to let you know when the controls are at appropriate positions.

To see snaplines in action, follow these steps:

1. **Drag a button control from the Visual Studio toolbox to the form design surface and don't release the button.**
2. **Move the mouse pointer to the lower-right corner of the form.**

   As you approach the corner, blue snaplines appear, as shown in Figure 1-7.

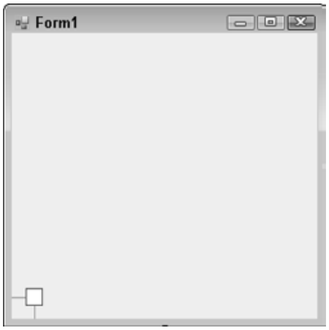


Figure 1-7: Snaplines provide visual cues for positioning controls.

3. **Drop the button on the form.**
4. **Drag a text box control to the form and don't release the text box.**
5. **Position the text box above the button and, when the snapline appears between the text box and the button, release the text box.**

Snaplines are visible any time you move controls on the design surface. The space between the button control and the edge of the form is the sum of the button's margin property and the form's padding property. Margins define the exterior space between controls, and padding designates space between a control's interior border and its contents. You can set these properties yourself or use snaplines to do it for you. See the section "Using the Properties Window," later in this chapter, for more information on setting control properties.

Locking controls

Getting the layout just right on a form can be a chore. You can keep your controls positioned where you want them by locking them in place. You have two options for locking controls:

- **To lock all your controls and the form itself,** choose Lock Controls from the Format menu.
- **To lock an individual control,** set the control's Locked property to True.

Setting the tab order

Nothing's worse than when you press the Tab key in a Windows application, and the cursor jumps to an unexpected field. The tab order determines the order in which the Tab button moves from field to field on your form. To set the tab order, follow these steps:

TIP

Good tab order is essential to creating an application that provides a high-quality user experience and one that is easy for everyone to use. This article at `http://msdn.microsoft.com/library/cb35a5fw.aspx` describes how to create an accessible application.

1. **Choose View➪Tab Order.**

   The controls on the page display numbers representing their tab order.

2. **Click to select the controls in sequential order to set the tab order, as shown in Figure 1-8.**

3. **Repeat Step 1 to take the form out of tab-order selection mode.**

To designate that a control shouldn't be tabbed to, set the control's `TabStop` property to `False`. Controls with the `TabStop` property set to `False` are skipped when the user presses Tab. Controls that aren't visible or enabled are also skipped when the user presses Tab.

To set a control as invisible or disabled, set the control's `Visible` and `Enabled` properties to `False`.

TIP

You can set a control's tab order by using the `TabIndex` property. (See the section "Using the Properties Window," later in this chapter, to see how to set a control's properties.)

Access a control's tasks with smart tags

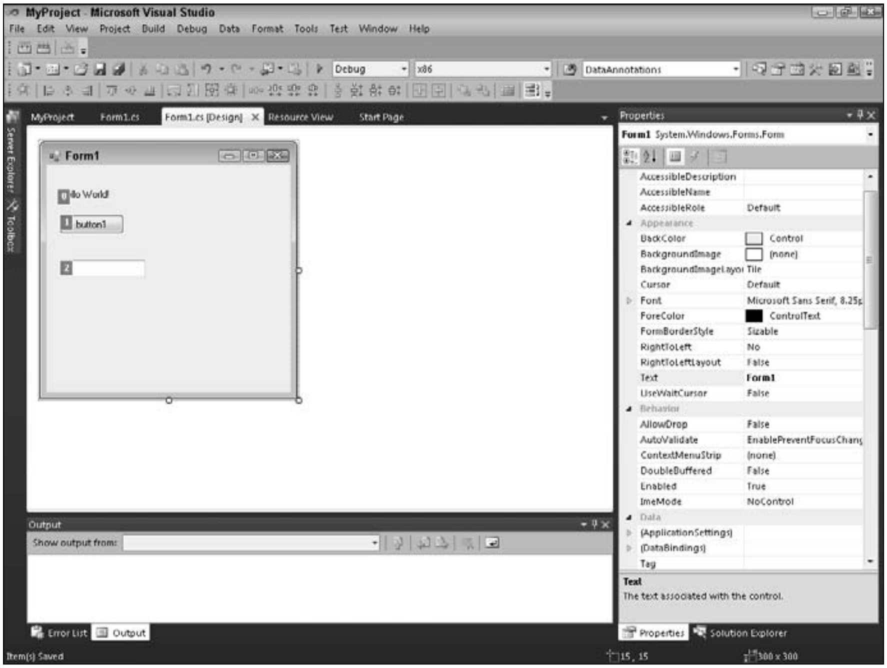
Some of the controls you add to your form can walk you through simple task wizards by using smart tags. Smart tag-enabled controls have a little arrow in the upper-right corner of the control. When you click the arrow, a list of common tasks associated with the control appears. To see a smart tag-enabled control in action

1. **Drag and drop a `CheckedListBox` control onto the forms designer.**

2. **Click the smart tag arrow to display a list of common tasks, as shown in Figure 1-9.**

3. **In the `CheckedListBox` Tasks window, click Edit Items.**

   From the String Collection Editor that appears, you can add items to the `CheckedListBox` control.



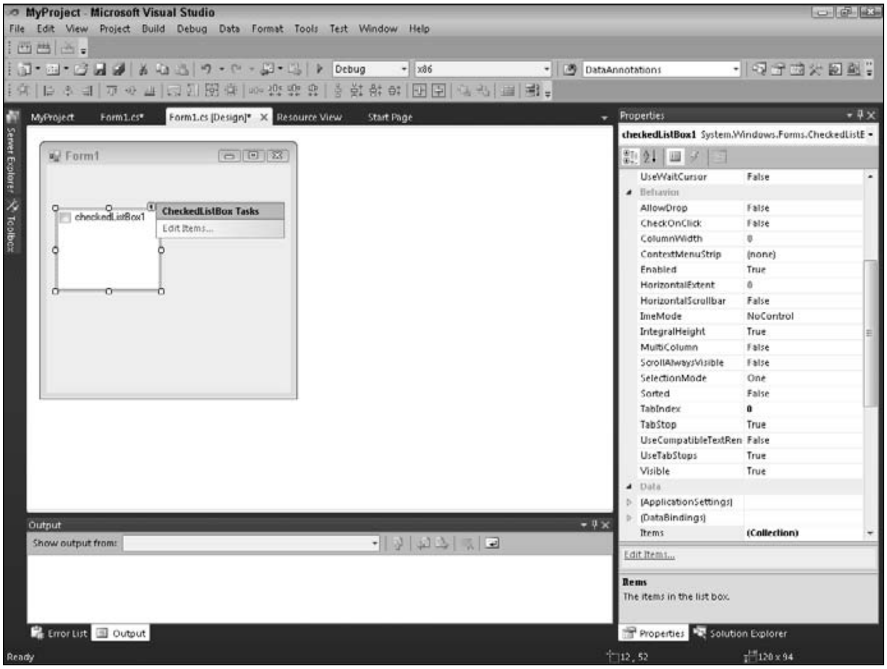
Properties

Form1 System.Windows.Forms.Form

AccessibleDescription	
AccessibleName	
AccessibleRole	Default
Appearance	
BackColor	Control
BackgroundImage	(none)
BackgroundImageLayou	Tile
Font	Microsoft Sans Serif, 8.25p
Cursor	Default
ForeColor	ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	**Form1**
UseWaitCursor	False
Behavior	
AllowDrop	False
AutoValidate	EnablePreventFocusChang
ContextMenuStrip	(none)
DoubleBuffered	False
Enabled	True
ImeMode	NoControl
Data	
(ApplicationSettings)	
(DataBindings)	
Tag	

Text
The text associated with the control.

Figure 1-8: Select the controls sequentially to set the tab order.



Properties

checkedListBox1 System.Windows.Forms.CheckedListE

UseWaitCursor	False
Behavior	
AllowDrop	False
CheckOnClick	False
ColumnWidth	0
ContextMenuStrip	(none)
Enabled	True
HorizontalExtent	0
HorizontalScrollbar	False
ImeMode	NoControl
IntegralHeight	True
MultiColumn	False
ScrollAlwaysVisible	False
SelectionMode	One
Sorted	False
TabIndex	**0**
TabStop	True
UseCompatibleTextRen	False
UseTabStops	True
Visible	True
Data	
(ApplicationSettings)	
(DataBindings)	
Items	**(Collection)**

Edit Items...

Items
The items in the list box.

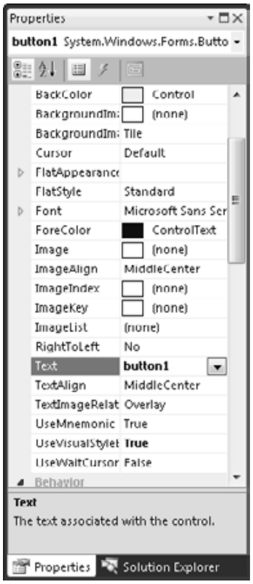
Figure 1-9: Click the control's smart tag arrow to display a list of common tasks.

The smart tags display a subset of tasks for the control. Use the control's Properties window to see all the tasks and properties available for the control.

Using the Properties Window

Every control has a set of properties and events that define how the control acts when your application runs. For example, the button control's `Text` property specifies the text that's displayed on the button. The button control has a `Click` event that's executed each time a user clicks the button.

The Visual Studio Properties window displays a control's properties and events in a grid, as shown in Figure 1-10.



BackColor	Control
BackgroundIm:	(none)
BackgroundIm:	Tile
Cursor	Default
FlatAppearanc	
FlatStyle	Standard
Font	Microsoft Sans Ser
ForeColor	ControlText
Image	(none)
ImageAlign	MiddleCenter
ImageIndex	(none)
ImageKey	(none)
ImageList	(none)
RightToLeft	No
Text	button1
TextAlign	MiddleCenter
TextImageRelat	Overlay
UseMnemonic	True
UseVisualStylet	True
UseWaitCursor	False
Behavior	

Text

The text associated with the control.

Figure 1-10: Setting a control's properties and events.

The Properties window is open by default. If you close the Properties window at some point, you have two options for opening it again:

- Choose View➪Properties Window
- Press F4

The Properties window has the following elements:

- **A drop-down list at the top displays the selected control.** Use the list to select a control to view.
- **A toolbar changes the view settings.** Your view-setting choices are, from left to right
  - *Categorized:* Properties appear in groups
  - *Alphabetical:* Properties are sorted in alphabetical order
  - *Properties:* The control's properties are displayed
  - *Events:* The control's events are displayed
- **Two columns display the name-value pairs of the control's properties or events, depending on the view setting.**
- **A brief description of the selected property or event appears at the bottom of the window.**

The Properties window is a visual tool that organizes and categorizes the thousands of properties and events available for manipulating controls. You use the Properties window for two tasks:

- Set a control's properties.
- Create event handlers for a control.

When you use the Properties window, Visual Studio generates the code to set the control's properties and create event handlers. You can do all these tasks by using code, although it's much easier to use Visual Studio. To read about how to set properties and work with events in code, see Book V, Chapter 4.

You can access thousands of properties and events by using the Properties window. Visual Studio groups properties and events into categories so that they're easier to manage. The following list describes the properties and events categories:

- `Action`: Creates handlers for events, such as `Click` and `Scroll`.
- `Accessibility`: Sets properties that make controls accessible to people with disabilities and makes the application easier to use for everyone.
- `Appearance`: Contains properties related to font, color, border, and cursor style. The `Text` property and text-formatting properties are in this group.
- `Behavior`: Contains properties and events related to how a control acts. Examples include `Sorted`, `TabStop`, and `SelectedIndexChanged`.

- `Data`: Sets a control's data source, gets items selected in a control, and responds to changes in the data.
- `Design`: Contains design-time properties, such as whether controls are locked. The `Name` property is here, too.
- `DragDrop`: Responds to events related to dragging and dropping controls.
- `Focus`: Captures events related to how the control gains and loses focus.
- `Key`: Responds to keyboard-related events.
- `Layout`: Sets layout properties, such as anchor, margin, and size.
- `Mouse`: Responds to mouse events, such as `MouseUp`.
- `WindowStyle`: Sets properties that define the Windows Form itself.

Within each control are properties that you can set. Here are some common properties that many controls share:

- `Name`: Sets the identifier used to reference the control in code.
- `Text`: Sets the text displayed in the control.

  To set the access key for a control, add an ampersand (&) in front of the letter to use as the access key in the `Text` property. For example, to set the letter P in Print as the access key, set the `Text` property for the control equal to `&Print`. A user can then press the Alt key in combination with the access key to execute the control.

Windows Forms are controls. You access the properties of a Windows Forms control the same way you access the properties for any control. Here are the common properties for Windows Forms:

- `Text`: Sets the text displayed on the form's title bar.
- `DefaultButton`: Sets the button that responds to the Enter key by default.
- `CancelButton`: Sets the button that responds to the Esc key by default.
- `MainMenuStrip`: Sets the primary menu control for the form.

In addition to the controls you use to customize your forms, you have events to consider. Events are actions that your application might handle or respond to. Examples of common events include

- `Click`
- `KeyDown`
- `Load`
- `ValueChanged`

You can get context-sensitive help for any property or event you see in the Properties window by positioning your cursor in the value field for the property or event, and pressing F1.

Setting properties

Properties are expressed as name-value pairs. A property's value isn't always as simple as `color=black`. Values can be complex. You can use the selectors, wizards, and ellipses on the Visual Studio Properties page to dynamically build complex property values.

Examples of complex properties include properties related to

- Setting data sources
- Editing collections of items
- Applying fonts and formatting
- Selecting images and files

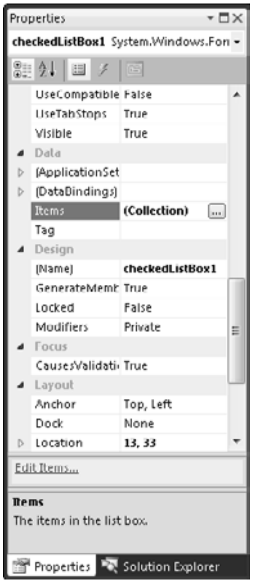
To set properties for a control on a Windows Form, follow these steps:

1. **Click the control in the forms designer.**
2. **Press F4 to display the Properties window.**

   The control's name and type appear on the drop-down list at the top of the Properties window.
3. **Click the Properties button on the Properties window's toolbar to display a list of the control's properties.**

   The Properties button is the third toolbar button from the left.
4. **Scroll down the properties list and enter property values.**

   You can type simple property values or select them from a drop-down list. Click the ellipsis button for more complex properties, such as collections to display a dialog box that builds the property, as shown in Figure 1-11. Click the plus sign (+) next to compound properties to set individual properties.



checkedListBox1 System.Windows.Fon

UseCompatible	False
UseTabStops	True
Visible	True
Data	
(ApplicationSet	
(DataBindings)	
Items	(Collection) ...
Tag	
Design	
(Name)	checkedListBox1
GenerateMemb	True
Locked	False
Modifiers	Private
Focus	
CausesValidati	True
Layout	
Anchor	Top, Left
Dock	None
Location	13, 33

Edit Items...

Items
The items in the list box.

Figure 1-11: Click the ellipsis button to build complex properties.

Responding to events

The Visual Studio Properties window displays a control's events. For example, when a user clicks a button, the button's `Click` event occurs. You can use event handlers to write code that fires each time the event fires.

To set an event handler by using the Properties window, follow these steps:

1. **Select a form's control by clicking the control.**
2. **In the Properties window, click the Events button to display the control's events.**

   To create an event handler by using a default name, either press Enter without typing a name or double-click the field for the name.

   Double-click a control on the Windows Forms Designer to create the control's default event handler.
3. **Type a name for the event handler next to the event and press Enter.**

   Visual Studio creates the event handler and opens the code editor to the event handler.

See Book V, Chapter 4 for more details about responding to events.

Chapter 2: Building Smart Client User Interfaces

In This Chapter

- **Customizing Windows interfaces**
- **Using dialog boxes**
- **Adding menus and toolbars**
- **Inheriting from a base Windows Form**
- **Laying out controls in a Windows Form**

Using Visual Studio to build Windows applications gives you a head start in your race against project deadlines. This chapter offers tools and techniques you can use to transform Visual Studio's out-of-the-box Windows projects into Windows applications that could make Bill Gates proud. In this chapter, you see most examples in VB and C#.

Building the Windows Container

As the container that holds your controls, the Windows Form is one of the most important elements in building Windows applications. The Windows Form is your application's foundation, and it's highly customizable.

The following sections explore some properties and controls that you can use to modify how your Windows Forms look and behave.

Setting common form properties

You can probably spot a typical Windows Form from ten paces:

- ✦ It usually has the familiar little red Close button in its upper-right corner.
- ✦ If it's like most windows, it has menus and toolbars.
- ✦ It's displayed on the taskbar along the bottom of the screen.

All these features of a typical window are determined by properties. Simply turning a property on or off can give a Windows Form a complete makeover. The default Windows Form that Visual Studio creates in all new Windows projects has most of these typical properties. You have to set some properties, such as menus, on your own.

Table 2-1 describes some common Windows Forms properties, sorted by property group.

Table 2-1 Common Windows Forms Properties

Property	*What It Does*
`FormBorderStyle`	Determines whether the window is resizable
`Text`	Sets the caption that appears on the form's title bar
`ContextMenuStrip`	Sets the shortcut menu that's displayed when a user right-clicks the form
`Name`	Sets the identifier used to access the form in code
`StartPosition`	Determines where the form appears when it's first opened
`AcceptButton`	Sets the default Enter button
`CancelButton`	Sets the default Esc button
`ControlBox`	Toggles the form's minimize, maximize, and close controls on and off
`IsMdiContainer`	Determines whether the form is a parent container in a Multiple Document Interface (MDI) application
`Icon`	Sets the icon displayed on the title bar and the taskbar when the form is minimized
`ShowInTaskBar`	Shows the form on the Windows taskbar
`MainMenuStrip`	Sets the form's menu control

You set these properties by using the Properties window. For more information on how to set a control's properties, see Chapter 1 of Book IV.

Creating dialog boxes

A dialog box is a Windows Form with attitude. The dialog box pops up on top of forms that are already open and refuses to leave until the user responds. Many Windows applications use dialog boxes to get a user's attention.

Most dialog boxes are *modal,* which means that they must be closed or hidden before users can continue working with other windows. Modeless forms can remain open alongside other windows. Modeless dialog boxes are harder to manage because you have to keep track of the different windows a user works with. Visual Studio provides a modal dialog form that you can add to your application.

To add a modal dialog box to an existing Windows project, follow these steps:

1. **Press Ctrl+Shift+A to open the Add New Item window.**
2. **Select the Dialog template.**
3. **Enter a name for the dialog box, such as MyDialog.**
4. **Click Add.**

   Visual Studio adds the dialog box to your project.

The difference between a modal and a modeless dialog box is the method with which you invoke it. For modal dialog boxes, you use the `ShowDialog` method after the dialog object has been created. For modeless dialogs, you use the `Show` method after the dialog object has been created.

The dialog box is just a regular Windows Form with the modified properties shown in Table 2-2.

Table 2-2 Modified Properties

Property	*Sets To*
`AcceptButton`	The form's OK button
`CancelButton`	The form's Cancel button
`FormBorderStyle`	`FixedDialog`
`MinimizeBox`	`False`
`MaximizeBox`	`False`
`ShowInTaskbar`	`False`
`StartPosition`	`CenterParent`

The dialog box also has an OK button and a Cancel button. The Cancel button's `DialogResult` property is set to `Cancel`. The `DialogResult` property returns the selected value to the parent form of the dialog box when the button is clicked.

To use the dialog box, follow these steps:

1. **Add a label and a text box to the dialog box you create in the preceding set of steps; set the Text property of the text box to `txtFromDialog` and set the Modifiers property to `Public` in the Properties view.**
2. **Set the OK button's `DialogResult` property to OK.**

   When the user clicks the OK button in the dialog box, the value set in the `DialogResult` property is sent to the parent form.
3. **Add a label, text box, and button to the project's parent form.**
4. **If no other forms exist in the project, press Ctrl+Shift+A to open the Add New Items window and add a Windows Form.**
5. **Double-click the button you created in Step 3 to access the button's `Click` event.**

   The Code Editor appears.
6. **In the button's `Click` event, type this code:**
   - **VB**

     ```
     Dim dlg As New MyDialog

     If dlg.ShowDialog() = DialogResult.OK Then
        Me.TextBox1.Text = dlg.txtFromDialog.Text
     End If
     ```

   - **C#**

     ```
     MyDialog dlg = new MyDialog();

     if (dlg.ShowDialog() == DialogResult.OK)
     {
        this.textBox1.Text = dlg.txtFromDialog.Text;
     }
     ```

   The first two lines of code open the dialog box you create in the preceding set of steps. The remaining four lines test the `DialogResult` property of the dialog box and set the `Text` property on the text box of the parent form.
7. **Press Ctrl+F5 to run the application.**

To test the application, follow these steps:

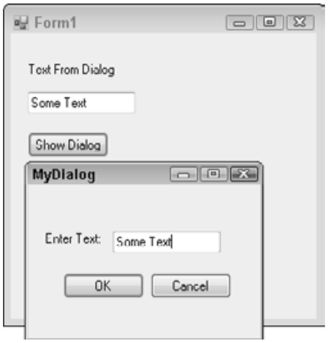
1. **On the parent form, click the button that launches the dialog box.**

   The dialog box appears.
2. **In the dialog box, type** Some Text **in the text box and then click OK.**

   The dialog box closes.

3. **The phrase *Some Text* appears in the text box in the parent form.**

   Figure 2-1 shows you the parent form with the dialog box open.



Text From Dialog

Some Text

Enter Text: Some Text

Figure 2-1: Text typed in the dialog box appears in the parent form.

ON THE WEB

You can download the complete program (both C# and VB versions) for this example from this book's companion Web site at `www.dummies.com/go/vs2010`.

Visual Studio has a number of preconfigured dialog-box components that you can use in your applications. You work with the dialog-box components in the Visual Studio toolbox by using code (unlike working with the dialog box you create in this section). The preconfigured dialog boxes create a consistent way for you to provide access to common features, such as printing and opening files. Table 2-3 lists the preconfigured dialog boxes.

Table 2-3 Preconfigured Dialog Boxes

Toolbox Tab	*Dialog Box*	*What It Does*
Dialogs	`ColorDialog`	Displays a color palette and controls for selecting a color
	`FolderBrowserDialog`	Prompts user to select a folder
	`FontDialog`	Prompts user to select a font
	`OpenFileDialog`	Prompts user to open a file
	`SaveFileDialog`	Prompts user to save a file
Printing	`PrintDialog`	Prompts user to select a printer and configure settings
	`PageSetupDialog`	Prompts user to change page-related settings
	`PrintPreviewDialog`	Previews the document to be printed

Use the preconfigured dialog boxes, rather than create your own. The steps for using a preconfigured dialog box vary slightly because each dialog box has its own set of properties you must set.

To use the `ColorDialog` dialog box, follow these steps:

1. **Create a new Windows Form.**
2. **Drag and drop a text box and button onto the form.**
3. **Drag and drop a `ColorDialog` control onto the form from the Dialogs tab of the toolbox.**

   The `ColorDialog` appears at the bottom of the Windows Forms Designer, rather than on the Windows Form.
4. **Set the `ColorDialog` control's `Name` property to `MyColorDialog`.**
5. **Double-click the button on the Windows Form.**

   The Code Editor appears.
6. **Type the following code in the Code Editor:**

   - **VB**

     ```
     If MyColorDialog.ShowDialog() = Windows.Forms.DialogResult.OK Then
         txtSelectedColor.BackColor = MyColorDialog.Color
     End If
     ```

   - **C#**

     ```
     if( MyColorDialog.ShowDialog() == DialogResult.OK )
     {
         txtSelectedColor.BackColor = MyColorDialog.Color;
     }
     ```

   Use the `ShowDialog()` method to display preconfigured dialog boxes.
7. **Press Ctrl+F5 to run the application.**
8. **Click the button on the form to launch the `ColorDialog` control.**

   The `ColorDialog` appears, as shown in Figure 2-2.
9. **Select a color and then click OK.**

   The background color of the text box changes.

You can download the complete program (both C# and VB versions) for this example from this book's companion Web site at `www.dummies.com/go/vs2010`.

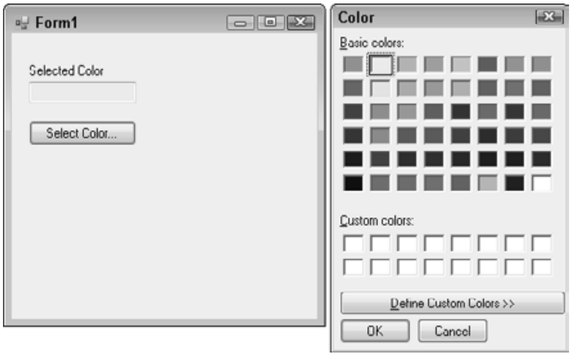


Figure 2-2: This control is a preconfigured dialog box.

Adding menus and toolbars

Most Windows applications have menus and toolbars that give users access to commands. Visual Studio provides these two menu controls and three toolbar controls:

- `ContextMenuStrip`: The shortcut menu that appears when a user right-clicks
- `MenuStrip`: A standard menu that usually appears at the top of a form
- `StatusStrip`: A band that usually appears along the bottom of a form that displays status feedback information
- `ToolStrip`: A control that creates toolbars
- `ToolStripContainer`: A container for hosting menu and toolbar controls

The `ToolStrip` control, the granddaddy of all the menu and toolbars controls, has these features:

- Items that you can add to the menu and toolbar controls, such as `ToolStripMenuItem`
- Container controls that allow menus and toolbars to coexist in the same container
- Properties that allow you to set the look and feel of menus and toolbars

Because all menu and toolbar controls are related, they share the `ToolStrip` control's features. The procedure for adding items, working with containers, and setting properties is consistent across all menu and toolbar controls.

To add a menu to a form and configure the menu, follow these steps:

1. **Drag a `MenuStrip` control to a form and drop it.**

   The menu docks itself to the top of the form.

2. **Click the arrow in the upper-right corner of the `MenuStrip` control to display a list of tasks.**

3. **In the MenuStrip Tasks dialog box, click Insert Standard Items.**

   The `MenuStrip` control adds menu items for File, Edit, Tools, and Help, as shown in Figure 2-3.

4. **Repeat Steps 1–3 to add a `ToolStrip` control to the form and insert standard items on the toolbar.**

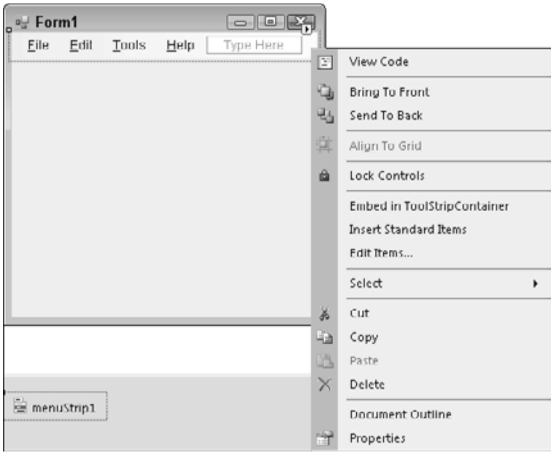


Figure 2-3: Click Insert Standard Items to add standard menu items to the `MenuStrip` control.

All the menu and toolbar controls are smart tag-enabled. A smart tag-enabled control has a little arrow in its upper-right corner. Click this arrow to access a task dialog box that displays common tasks for each control. The `MenuStrip` and `ToolStrip` controls display the same task dialog box. (Figure 2-3 shows the MenuStrip Tasks dialog box.) The available `MenuStrip` and `ToolStrip` tasks are shown in this list:

- **Embed in ToolStripContainer:** Moves the control inside a ToolStripContainer control
- **Insert Standard Items:** Adds standard command items to the control
- **RenderMode:** Sets a specific style for the control

- **Dock:** Sets the control's location to the top or bottom, or the left or right side, of the form
- **GripStyle:** Makes the control's move handle hidden or visible
- **Edit Items:** Opens the Items Collection Editor, which you use to edit the control's items

Menus and toolbars display commands that users can execute. Each command is an item on the menu or toolbar control. A control's entire set of commands is the control's items collection. You can edit a control's items collection by using the Windows Forms Designer or the Items Collection Editor.

To edit items by using the Windows Forms Designer, follow these steps:

1. **Drag and drop a `StatusStrip` control on the form.**

   The control docks itself to the bottom of the form by default.

2. **Click the drop-down arrow in the control.**

   A list of available `StatusStrip` items appears, as shown in Figure 2-4.

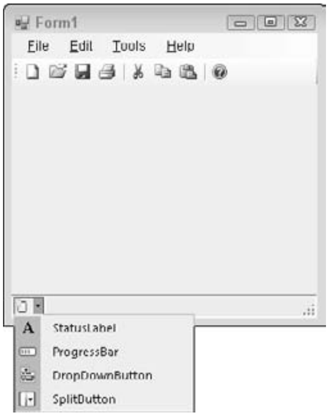


Figure 2-4: Select an item from the drop-down list to add to the `Status Strip` control.

3. **Select a `StatusLabel` control from the list.**

   The `StatusLabel` control is added to the `StatusStrip` control.

4. **Repeat Steps 2 and 3 to add a progress bar.**

To add items to a `MenuStrip` control by using the Windows Forms Designer, follow these steps:

1. **Click the `MenuStrip` control in the form.**

   A command placeholder named `MenuItem`, with the prompt Type Here, appears; see Figure 2-5.

   To insert a command between existing commands, right-click a command and choose Insert from the shortcut menu. Choose an item to insert from the list of available items.

2. **Type a command name, such as `View`, and then press Enter.**

   The placeholder expands for you to add more commands.

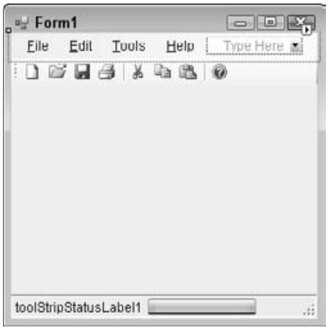


Figure 2-5: Type a command in the `MenuItem` placeholder.

Table 2-4 lists the items you can add to the menu and toolbar controls.

Table 2-4 Items You Can Add to the ToolStrip Controls

Control Name	*Item You Can Add*
`MenuStrip`	`MenuItem`
	`ComboBox`
	`Separator`
	`TextBox`
`ToolStrip`	`Button`
	`Label`
	`SplitButton`
`ToolStrip`	`DropDownButton`
	`Separator`
	`ComboBox`
	`TextBox`
	`ProgressBar`

Control Name	*Item You Can Add*
`StatusStrip`	`StatusLabel`
	`ProgressBar`
	`DropDownButton`
	`SplitButton`

You can start a control's Items Collection Editor in one of three ways:

- From the control's Tasks dialog box
- By using the control's Properties window
- From the control's shortcut menu

To use the Items Collection Editor from the control's shortcut menu:

1. **Right-click the form's `StatusStrip` control.**

   The control's shortcut menu appears.

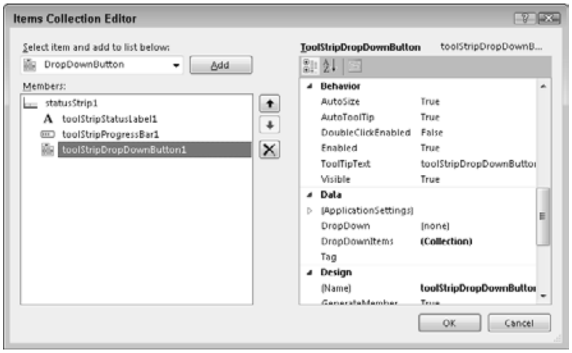
2. **Choose Edit Items.**

   The Items Collection Editor appears.

3. **Click the drop-down list at the top of the editor to display a list of available items for the control.**
4. **Select `DropDownButton` from the list.**
5. **Click Add.**

   A `DropDownButton` control is added to the members list, as shown in Figure 2-6.

6. **Click the up arrow to move the `DropDownButton` control to the top of the list.**



Behavior
AutoSize True
AutoToolTip True
DoubleClickEnabled False
Enabled True
ToolTipText toolStripDropDownButtor
Visible True
Data
(ApplicationSettings)
DropDown (none)
DropDownItems (Collection)
Tag
Design
(Name) toolStripDropDownButtor
GenerateMember True

Figure 2-6: Click Add to add an item to the control's items collection.

7. **Select the progress bar control from the Members list.**

   The progress bar control is named `ToolStripProgressBar1`.

8. **Click the button with the X on it to delete the progress bar.**

9. **Click OK.**

   The Items Collection Editor closes.

TIP

The properties of individual items on menu and toolbar controls are easier to manage if you use the Items Collection Editor instead of the Properties window.

Using a ToolStripContainer

The `ToolStripContainer` control makes it easy to host multiple menu and toolbar controls on a single form. The `ToolStripContainer` control has panels, as shown in Figure 2-7, where you place your menu and toolbar controls. The advantage of using `ToolStripContainer` is that your controls automatically stack horizontally and vertically in the panel.

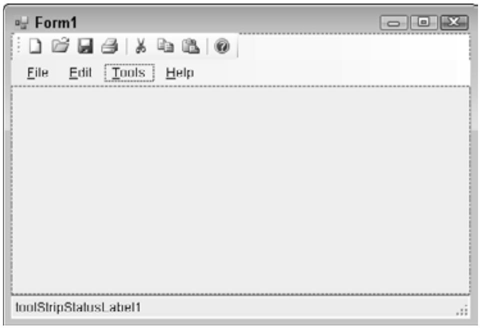


Figure 2-7: Place menu and toolbar controls in this control's panels.

To use a `ToolStripContainer` control in a Windows Form, follow these steps:

1. **Drag and drop a `ToolStripContainer` control on a Windows Form.**

2. **Click Dock Fill in Form in the `ToolStripContainer` control's Tasks dialog box.**

   The control expands to fill the form. The control's top panel is visible by default.

3. **Drag and drop a `MenuStrip` control on the top panel.**
4. **Click the arrow tab on the `ToolStripContainer` control's top pane.**

   The pane expands to accommodate another menu or toolbar.
5. **Drag and drop a `ToolStrip` control on the top panel, positioned below the `MenuStrip` control.**
6. **Click the tab on the bottom panel to make the panel visible.**
7. **Add a `StatusStrip` control to the bottom panel.**

Users can use the `ToolStripContainer` control to customize their work environments by moving toolbars and menus around within the panels of the `ToolStripContainer` control. Figure 2-8 shows you an example of a toolbar being moved.

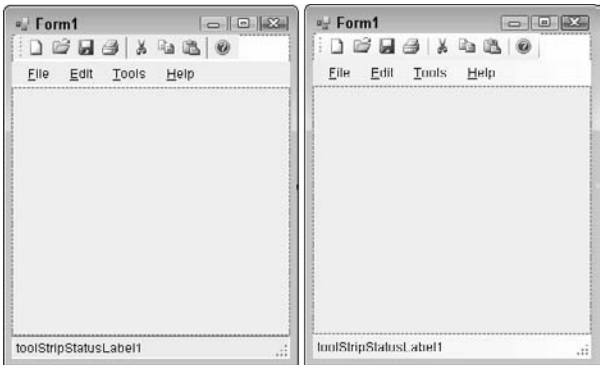


Figure 2-8: Using a control to move toolboxes.

Controlling styles

You can style the menu and toolbar controls by using the `RenderMode` property. You can use it to specify how you want Windows to draw the menu or toolbar. The .NET built-in styles render your menus and toolbars to look like Office or Windows. You can set the `RenderMode` property to one of these four values:

- `Custom`: Sets the style to a custom renderer
- `ManagerRenderMode`: Uses the renderer specified in the `ToolStripManager`
- `Professional`: Uses a style that looks like Office 2003 and Windows XP
- `System`: Uses a flat Windows style

Figure 2-9 shows the same form with two different rendered styles. The form on the left uses system styling, and the form on the right uses professional styling. The professionally styled menus and toolbars are rounded and more elegant. You can set these styles by using the Tasks dialog box for each of the controls. The most obvious difference is the use of hot keys.

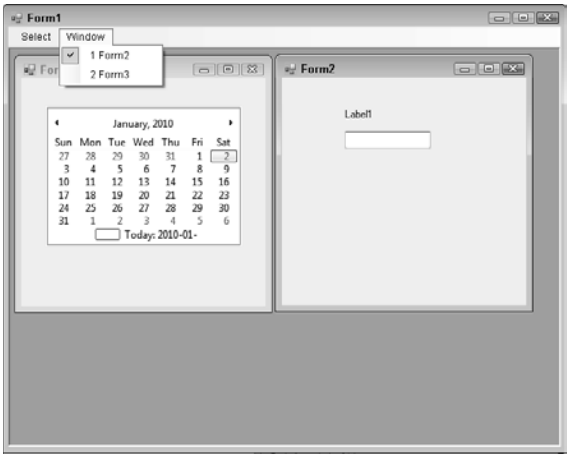


Figure 2-9: The System menu style is "flatter" than the Professional style.

You use `ToolStripManager` to set a single style for all `ToolStrip` controls. For example, to set all the menus and toolbars to Professional, follow these steps:

1. **Set the `RenderMode` property for each menu and toolbar control to `ManagerRenderMode`.**
2. **Set the `RenderMode` property on `ToolStripManager` in the form's Load event by using this code:**
   - **VB**

     ```
     ToolStripManager.RenderMode = ToolStripManagerRenderMode.Professional
     ```

   - **C#**

     ```
     ToolStripManager.RenderMode = ToolStripManagerRenderMode.Professional;
     ```

   You access `ToolStripManager` in code. It's not a control that you can access from the toolbox, like on other menus and toolbars.

Creating a Multiple Document Interface

You may have used a Windows application where all the application windows are contained within a parent window. An application in which all child windows are contained within a parent window is a Multiple Document Interface (MDI).

Creating an MDI application involves these high-level steps:

1. **Create the parent container form.**
2. **Create child forms.**
3. **Write code that opens the child forms in the container.**

To create the parent MDI form, follow these steps:

1. **Create a new Windows application in Visual Basic.**

   Visual Studio creates a new Windows project with an empty Windows Form named Form1.

   See Chapter 1 in Book III for more information about creating Windows applications.
2. **Set the Form1 `IsMdiContainer` property to `True`.**
3. **Add a `MenuStrip` control to the form.**
4. **Add the menu items Select and Window.**
5. **Add the menu items Open Form 2 and Open Form 3 to the Select menu.**

After you have the parent form, create these child forms to open in the parent MDI form:

1. **Add a new form named Form2.**
2. **Add a label and a text box to the form.**
3. **Add a new form named Form3.**
4. **Add a `MonthCalendar` control to the form.**

In the parent MDI container Form1, follow these steps:

1. **Double-click the form to access the form's `Load` event.**
2. **Above the `Load` event, type this line:**

   ```
   Dim frmChild As Form
   ```
3. **In the `Load` event, type this line:**
   - **VB**

     ```
     MenuStrip1.MdiWindowListItem = WindowToolStripMenuItem
     ```
   - **C#**

     ```
     MenuStrip1.MdiWindowListItem = WindowToolStripMenuItem;
     ```

This line sets the Windows menu item as the MdiWindowListItem control for the menu strip. As child windows are opened, the Windows menu displays the windows, along with a check mark, next to the active window.

4. **Below the `Load` event, type these lines:**

   - **VB**

   ```
   Private Sub GetChild(ByRef frmChild As Form)
      frmChild.MdiParent = Me
      frmChild.Show()
   End Sub
   ```

   - **C#**

   ```
   private void GetChild(ref Form frmChild)
   {
        frmChild.MdiParent = this;
        frmChild.Show();
   }
   ```

   This code sample creates a procedure named GetChild that accepts a child form as a parameter. When the child form is passed to GetChild, the procedure sets the child form's MDI parent container as the displayed form and displays the child form in the container.

5. **In the Windows Forms Designer, double-click the Open Form 2 menu item.**

   The Click event is created.

6. **Type this line in the `Click` event:**

   ```
   GetChild(New Form2)
   ```

   This line passes Form2 to the GetChild procedure.

7. **Repeat Steps 4 and 5 for the Form 3 menu item and substitute Form3 for Form2 in the code.**

The entire code sample (minus the hidden code that Visual Studio creates) for Form1 is shown in Listing 2-1.

Listing 2-1: Code That Services the Multiple Document Interface

```
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles MyBase.Load
        MenuStrip1.MdiWindowListItem = WindowToolStripMenuItem
    End Sub
```

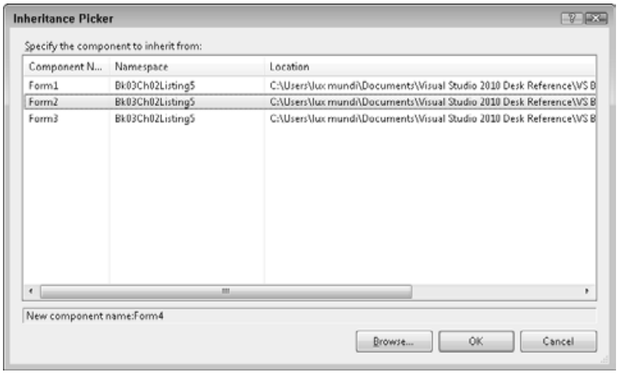
```
    Private Sub GetChild(ByRef frmChild As Form)
        frmChild.MdiParent = Me
        frmChild.Show()
    End Sub

    Private Sub OpenForm2ToolStripMenuItem_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles OpenForm2ToolStripMenuItem.Click
        GetChild(New Form2)
    End Sub

    Private Sub OpenFormToolStripMenuItem_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles OpenFormToolStripMenuItem.Click
        GetChild(New Form3)
    End Sub
End Class
```

To test the MDI application, follow these steps:

1. **Press Ctrl+F5 to run the application.**
2. **Choose the Select menu item.**
3. **Choose Open Form 2 from the Select menu.**

   Form2 opens and is contained within the parent form.
4. **Repeat Steps 1-3 to open Form3.**
5. **Click the Windows menu.**

   The Windows menu displays the open windows. A check mark appears next to the active window, as shown in Figure 2-10.



Specify the component to inherit from:

Component N...	Namespace	Location
Form1	Bk03Ch02Listing5	C:\Users\lux mundi\Documents\Visual Studio 2010 Desk Reference\VS B
Form2	Bk03Ch02Listing5	C:\Users\lux mundi\Documents\Visual Studio 2010 Desk Reference\VS B
Form3	Bk03Ch02Listing5	C:\Users\lux mundi\Documents\Visual Studio 2010 Desk Reference\VS B

New component name:Form4

Figure 2-10: Use the Windows menu to manage child windows.

Taking advantage of visual inheritance

Sometimes, you want to reuse forms. You can inherit from an existing form rather than create forms from scratch. You need two forms: the base form and one to inherit from the base form.

To create the base form, follow these steps:

1. **Open the Add New Item window and add a new Windows Form to the project.**

   You can press Ctrl+Shift+A to open the Add New Item window.

2. **Add a label and a text box to the form.**
3. **Choose Build⇨Build Solution.**

You must build the base form in order to inherit from it.

To inherit from the base form, follow these steps:

1. **Choose Project⇨Add New Item.**

   The Add New Item window appears.

2. **Click the Inherited Form template.**
3. **Enter a name for the form.**
4. **Click Add.**

   The Inheritance Picker appears.

5. **Click the base form in the Inheritance Picker.**
6. **Click OK.**

   The inherited form displays the base form's controls.

You must build your solution whenever you make changes to the base form in order to make changes appear in the inherited form.

Laying Out Your Controls

As UI design becomes more sophisticated, users expect to have more control over the user experience. This means being able, at minimum, to resize windows. Resizing can wreak havoc on the controls that you spend lots of time nudging into their proper positions. Windows Forms provide several options for managing controls for resizing, including containers for grouping controls and properties for locking controls in place.

Grouping controls

The Visual Studio toolbox has a group of container controls. Use container controls whenever you need to

- Easily manipulate groups of controls in the Windows Forms Designer
- Create visible groupings of controls
- Make it easier to resize a form

The container controls are shown in this list:

- `FlowLayoutPanel`: Arranges controls in a horizontal or vertical flow
- `GroupBox`: Creates a visible grouping of controls with a caption
- `Panel`: Creates a scrollable grouping of controls
- `SplitContainer`: Arranges controls on two separate panels separated by a movable bar
- `TabControl`: Arranges controls on multiple tabs or pages, similar to a file folder
- `TableLayoutPanel`: Arranges controls in a tabular grid

You can position container controls inside other controls. The `FlowLayoutPanel` and `TableLayoutPanel` controls can dynamically reposition the child controls placed inside them when the Windows Form is resized. To see `FlowLayoutPanel` in action, follow these steps:

1. **Add a `FlowLayoutPanel` control to a form.**
2. **Click the arrow in the upper-right corner of the control.**

   The control's task dialog box appears.
3. **Click `Dock` in the parent container.**

   The `FlowLayoutPanel` control expands to fill the form.
4. **Add two text boxes to the form.**
5. **Grab the second text box with your mouse and try to move the box below the first one.**

   The text box "jumps" back because the controls placed in a `FlowLayoutPanel` flow either horizontally or vertically.
6. **Add a `GroupBox` control to the form.**
7. **Place a label and a text box inside the `GroupBox` control.**

   As you add the controls, notice that snaplines appear inside the `GroupBox` control.

   Although you can position all the controls placed inside the `GroupBox` exactly where you want them, `GroupBox` flows with the rest of the controls in the `FlowLayoutPanel` control.

8. **Add a `MonthCalendar` control to the form.**
9. **Press Ctrl+F5 to run the form.**
10. **Resize the form and notice that the controls move.**

Figure 2-11 shows the form in its default size and again resized. Notice that the controls are stacked in the form on the left because the form is narrow. As the form is resized, the controls move to fill the form.

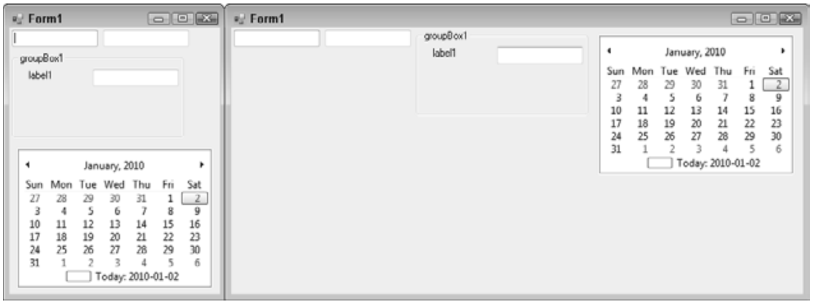


Figure 2-11: Resizing the window causes the controls to move.

You have some options for controlling the flow in a `FlowLayoutPanel` control:

- **To force a control to move to the next line:** Set the control's `FlowBreak` property to `True`.
- **To change the direction of the FlowLayoutPanel control:** Set the `FlowDirection` property.
- **To set FlowLayoutPanel so that child controls are clipped rather than wrapped to the next line:** Set the `WrapContents` property to `False`. Figure 2-12 shows a form that has `FlowLayoutPanel` with a clipped `MonthCalendar` control.

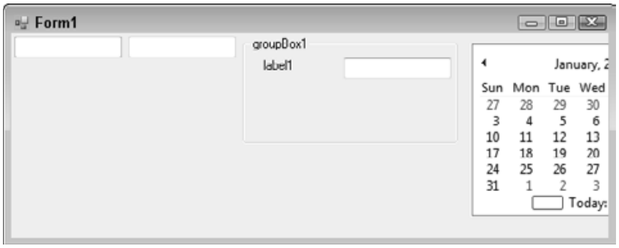


Figure 2-12: Set a property to `False` to clip controls rather than wrap them.

Setting layout properties

You may want to position controls in a way that they stay in place in the face of resizing. You can set many properties for controlling a control's layout:

- `Anchor`: Specifies an anchor position that the control maintains when the form is resized
- `Dock`: Positions the control so that one edge of the control is always touching the parent control's edge
- `AutoSize`: Allows a control to grow or shrink automatically
- `Margin`: Sets spacing outside a control's borders
- `Padding`: Adds spacing between a control's borders and its interior contents

The best way to picture these properties in action is to see them at work. The following sections walk you through using these properties.

Anchoring and docking

Anchoring and docking are two properties you can use to position a control when a form is resized. When you set a control's `Anchor` and `Dock` properties, you specify the edges of a form — top, bottom, left, right — to which you want to position your control. The `Dock` property accepts a `Fill` value, which forces the control to expand to touch all four sides of a form.

The primary difference between `Anchor` and `Dock` is that anchoring allows you to maintain a set distance between the edge of the control and the edge of the form. With docking, the control always maintains constant contact with the form's edge. There's no space between the control and the edge. Figure 2-13 shows a button that's anchored to the lower-left corner of the form and a status strip that's docked to the bottom.

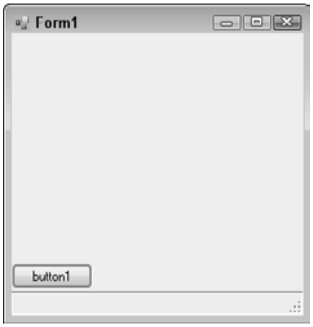


Figure 2-13: `Anchor` and `Dock` properties let you position a control.

Menus, toolbars, and status strips are always docked by default.

To anchor a Submit button to the lower-left corner of a form, follow these steps:

1. **Add a button to a form.**

   For more information on adding a button to a form, see the section in Chapter 1 of this mini-book about adding controls to your form.

2. **Click the button and press F4 to open the Properties window.**

3. **Scroll to the `Anchor` property.**

   The `Anchor` property is in the `Layout` category.

4. **Click the arrow on the drop-down list for the `Anchor` property.**

   A visual positioning tool appears.

5. **Click the left and bottom bars to set the anchor.**

6. **Click the top bar to clear the anchor.**

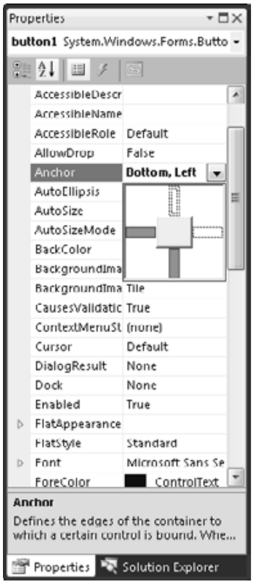
   Figure 2-14 shows you an example.

7. **Press Enter to set the property.**

8. **Press Ctrl+F5 to run the form.**

9. **Resize the form from the bottom, top, left, and right.**

   Notice that the button maintains its distance from the bottom and the left.



AccessibleDescr	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
Anchor	Bottom, Left
AutoEllipsis	
AutoSize	
AutoSizeMode	
BackColor	
BackgroundIma	
BackgroundIma	Tile
CausesValidatic	True
ContextMenuSt	(none)
Cursor	Default
DialogResult	None
Dock	None
Enabled	True
▷ FlatAppearance	
FlatStyle	Standard
▷ Font	Microsoft Sans Se
ForeColor	■ ControlText

Anchor

Defines the edges of the container to which a certain control is bound. Whe...

Figure 2-14: Click the bars to set and clear the anchor positions.

You can use the Anchor property to expand a control as the form expands. To set a text box to grow as the sides of a form expand, follow these steps:

1. **Add a text box to the center of the form.**
2. **Set the text box's Anchor property to Left, Right.**
3. **Press Ctrl+F5 to run the form.**
4. **Resize the form to the left and the right.**

Figure 2-15 shows the form in its default size and resized. Notice how the text isn't completely displayed in the text box. After the form is resized, the text box expands and maintains equal distance from the left and right sides.

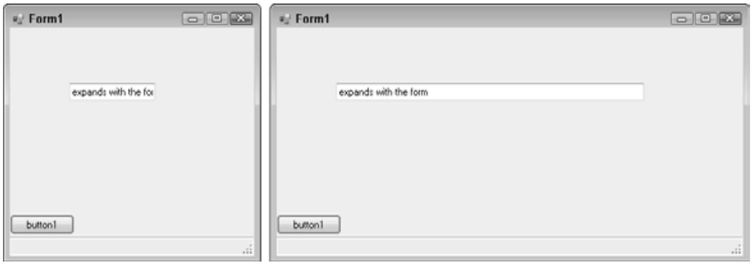


Figure 2-15: The text box expands as its anchor sides expand.

Setting a control's Dock property is similar to setting the Anchor property. With the Dock property, you specify the edges to which you want to dock the control. The control always maintains contact with the edge you specify. Figure 2-16 shows a form with a StatusStrip control docked on the bottom of the form. As the form is resized, the StatusStrip control remains at the bottom.

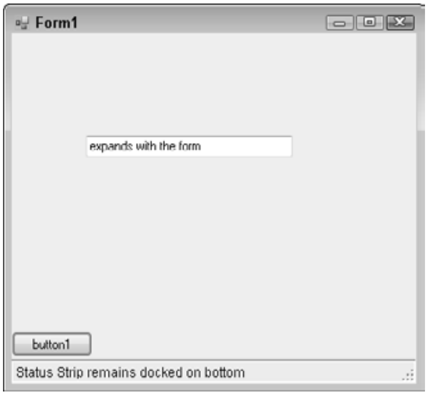


Figure 2-16: The control remains docked to the form's bottom edge as the form is resized.

Using automatic sizing

Two properties are related to setting automatic sizing:

- `AutoSize`: Specifies, by using a `True/False` value, whether the control should be automatically sized
- `AutoSizeMode`: Sets a control to `GrowAndShrink` or `GrowOnly`

The `AutoSizeMode` property works only if `AutoSize` is set to `True`. Although not all controls have an `AutoSizeMode` property, it provides more control than `AutoSize` when it's available. The size of a control grows rightward and downward.

To set the automatic sizing properties for a button, follow these steps:

1. **Add a button to a form.**
2. **Use the Properties window to set the button's `Text` property to `Please click this button`.**

   Notice that the button displays only part of the text.
3. **Set the button's `AutoSize` property to `True`.**

   The button expands to display the text, as shown in Figure 2-17.

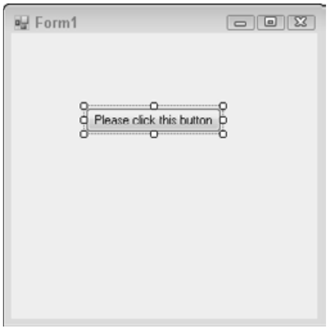


Figure 2-17: The button expands to display the text when the `AutoSize` property is set to `True`.

The `AutoSize` and `AutoSizeMode` properties honor the `MinimumSize` and `MaximumSize` property settings.

Setting margins and padding

Margins and padding set the space outside and within a control's borders, respectively. Because a control's border has four sides, the margins and padding properties comprise a set of properties. When you set the margins and padding properties, you can specify values for top, left, right, bottom, or all.

To set a control's margins and padding, follow these steps:

1. **Add a button to a form.**

   For more information about adding a button to a form, see the section in Chapter 1 of Book III about adding controls to your form.

2. **Click the new button and press F4 to open the Properties window.**
3. **Scroll to the `Margin` property.**
4. **Click the plus (+) sign next to the `Margin` property to expand the property.**
5. **In the `All` property, type** 20.

   All the button's margins are now set to 20.

6. **Drag another button to the form.**

   As you approach the first button, notice that the snaplines are farther apart than usual because the first button's margin is higher.

   The space between the two buttons when the snaplines are present is 23. The snapline's distance between controls is a sum of the two control's margins. The default margin for a button control is 3.

7. **Allow the second button to snap to the first button and drop the second button.**
8. **Change the padding property on the second button to `All = 10`.**
9. **Resize the second button so that you can see the button's text.**

   Notice the padding that appears around the text now. Figure 2-18 shows you an example.

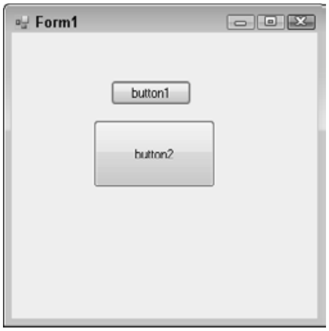


Figure 2-18: The second button's text is padded in all directions.

Chapter 3: Building Visual C++ Applications

In This Chapter

- **Taking a fresh look at C++**
- **Peeking at the Visual C++ application wizards**
- **Creating managed and unmanaged applications**
- **Referencing external libraries in your C++ projects**

Visual C++ has been around for quite some time, and C++, the core language on which Visual C++ is based, is even older! (C++ has been around since the mid-1980s and still exists in many legacy applications and even in new applications where performance is a factor.) Just because Visual C++ is, shall I say, mature doesn't mean that it can't hold its own against more modern programming languages. Many folks still use Visual C++ when they need to perform tasks for which performance is of utmost importance. Or, they also use Visual C++ to take advantage of the huge pile of C++ code they've developed over the years. Many developers prefer to develop managed code using C++ because of the ease of interoperability with existing unmanaged C++ code.

This chapter gives you the scoop on Visual C++ and shows you how to create Visual C++ applications by using Visual Studio 2010. Fortunately, Microsoft has made great strides in updating Visual C++ for standards conformance. The C++ standards committee is actively working on the new language features for the C++0x standard, which will offer exciting new features to the language.

Getting to Know Visual C++

Visual C++ is one language in the suite of .NET languages. Unlike the other .NET languages, the core syntax libraries in Visual C++ and its primary helper libraries aren't necessarily part of the .NET Framework. You can write managed Visual C++ applications, but most developers who write in C++ want to output native code. As a result, most applications written in Visual C++ don't require the .NET Framework.

Creating a full-blown Windows application by using just C++ syntax can be quite challenging. If you're guessing that this is where Visual C++ comes in, you're right. Visual C++ is different from plain old C++ because it provides additional libraries, tools, and widgets that you can use to build sophisticated applications for Windows and the Web.

Visual C++ includes these elements:

- **Standard libraries for building C++ applications:** The C++ object-oriented language was created to extend the popular C language. The downside of using a low-level language like C++ is that you have to write more code to get something done than you do when you're using a higher-level language, such as Visual Basic or C#.

  The result is that C++ programs are smaller and faster than higher-level languages because they can

  - Run without the .NET common language runtime
  - Interact directly with the operating system (OS)
  - Manage their own memory and resources

  C++ compilers exist for almost every OS. If you write your application by using only C++ syntax, you may be able to compile your program to run in Unix or Linux, even if you wrote it on a Windows computer.

- **Microsoft Foundation Classes (MFC), a library for building Windows applications:** The MFC library of classes wraps the Windows application programming interface (API). The MFC library was written in C++ to provide object-oriented access to the features and commands of the Windows API. Before MFC was created, developers used the procedural language C to access the Windows API.

- **The Active Template Library (ATL) for building small, lightweight components:** The ATL library of C++ classes is designed for building components such as ActiveX controls. Components built with ATL are usually consumed by another application. ATL Server is an extension of ATL for building powerful, high-performance Web applications and Web services.

- **The Standard Template Library (STL), a library for string handling, containers, algorithms, and streams:** The STL library has been around since the late 1990s and, in fact, became standardized in 1999 and is included with most C++ development environments.

  ATL creates components known as COM objects. The COM (Component Object Model) standard defines how components talk with each other. Components created with ATL conform to the COM standard and can use the COM services provided by Windows to communicate.

✦ **Support for accessing the services of the .NET Framework:** Earlier in this section, I said that C++ applications don't require .NET. That's right: Although they don't require .NET, they can access the services of .NET when they want. As a result, code written in Visual C++ is either

- *Managed:* A Visual C++ program that uses the services of the .NET Framework. Any code that's executed in the .NET Common Language Runtime (CLR) is managed. Programs written in Visual Basic and C# are always managed because they can't be executed without the services of .NET, such as the CLR. You can read more about CLR and .NET in Book I, Chapter 2.
- *Native code:* An application that doesn't require .NET. You create unmanaged Visual C++ applications by using the MFC library, the ATL library, or any of the standard C++ libraries. Even unmanaged Visual C++ applications can access some managed components through a mechanism that this chapter discusses next.

The ability of managed and unmanaged code to coexist peacefully in Visual C++ is interoperability, or Interop. To read more about Interop, search the Visual Studio Help index for Interop. The .NET framework uses Platform Invoke (P/Invoke) as a mechanism for managed code to execute unmanaged code. The MSDN article at `http://msdn.microsoft.com/en-us/library/2x8kf7zx(VS.80).aspx` explains P/Invoke in detail.

You can use Visual C++ to create all kinds of applications and components:

✦ Console applications

✦ Controls

✦ Dynamic link libraries (DLLs)

✦ Web services

✦ Internet Server Application Programming Interface (ISAPI) Filters

✦ Windows applications

✦ Windows services

Despite the rapid migration of Windows application development to the .NET languages such as C# and Visual Basic, developers in recent years have created rich libraries for C++ that allow developers to create applications with much less coding effort. An example is the Boost project (`www.boost.org`), a free, peer-reviewed set of libraries developed by some of the best C++ developers in the world. Boost contains libraries for such services as threads and thread pools, regular expressions, smart or memory-managed pointers, and sockets and streams.

Introducing Visual C++ Templates and Wizards

Because Visual C++ is a low-level language, you have to write more lines of code to do even simple tasks. Lots of code is involved in wiring up all the Visual C++ libraries in just the right way so that you can start building software. Thankfully, Visual Studio provides many templates and application wizards for building Visual C++ applications.

Visual Studio provides project templates for managed C++ applications. Because unmanaged applications require more code, Visual Studio uses application wizards to walk you through the process of setting up your application. Table 3-1 lists the project templates for managed applications, and Table 3-2 lists the Visual C++ application wizards.

Table 3-1 Visual Studio Project Templates for Creating Managed Visual C++ Applications

Project Template	*What It Does*
ASP.NET Web Service	Creates XML Web services by using ASP.NET
Class Library	Creates reusable class libraries and components that you can use in other projects
CLR Console Application	Creates a command line application without a graphical user interface (UI)
CLR Empty Project	Creates an empty project to which you must add files manually
SQL Server Project	Creates a data access class library that you can deploy to SQL Server
Smart Device Project	Creates a project aimed at smart devices, such as a BlackBerry
Windows Forms Application	Creates a Windows application
Windows Forms Control Library	Creates custom controls for use in Windows Forms
Windows Service	Creates an application that runs as a Windows service

Table 3-2 Visual Studio Application Wizards for Creating Unmanaged Visual C++ Applications

Project Template	*Application Wizard*	*What the Application Creates*
ATL Project	ATL Project Wizard	DLLs, applications, and services for creating components using ATL libraries
ATL Server Project	ATL Server Project Wizard	A Web application and ISAPI extension DLLs for responding to HyperText Transfer Protocol (HTTP) requests
ATL Server Web Service	ATL Server Project Wizard	An ATL server application configured to run as a Web service
ATL Smart Device Project	ATL Smart Device Project Wizard	A DLL or an application created by using ATL and designed to run on a smart device
Custom Wizard	Custom Application Wizard	A custom wizard
Makefile Project	Makefile Application Wizard	A project that sets build settings for a project built by using the command line
MFC ActiveX Control	MFC ActiveX Control Wizard	An ActiveX control using the MFC library
MFC Application	MFC Application Wizard	A Windows application using the MFC library
MFC DLL	MFC DLL Wizard	A DLL using the MFC library
Win32 Console Application	Win32 Application Wizard	A command line application with or without MFC and ATL libraries
Win32 Project	Win32 Application Wizard	A Windows or console application, DLL, or class library with or without MFC and ATL libraries

The ATL, MFC, and Win32 application wizards include some support for smart devices, but more specific support is available if you create a Smart Device Project.

Visual C++ projects are organized into three folders:

- **Header Files:** Holds source files that reference entities in the C++ libraries
- **Resource Files:** Holds resources, such as bitmap files and cursors, as well as the files that manage an application's resources
- **Source Files:** Holds source code files, including the main C++ source files that have the extension `.cpp`

The Visual Studio project templates and application wizards generate the header, source, and resource files for the kind of application you create. For more information, search for file types in the Visual Studio Help index.

Saying "Hello, Visual C++"

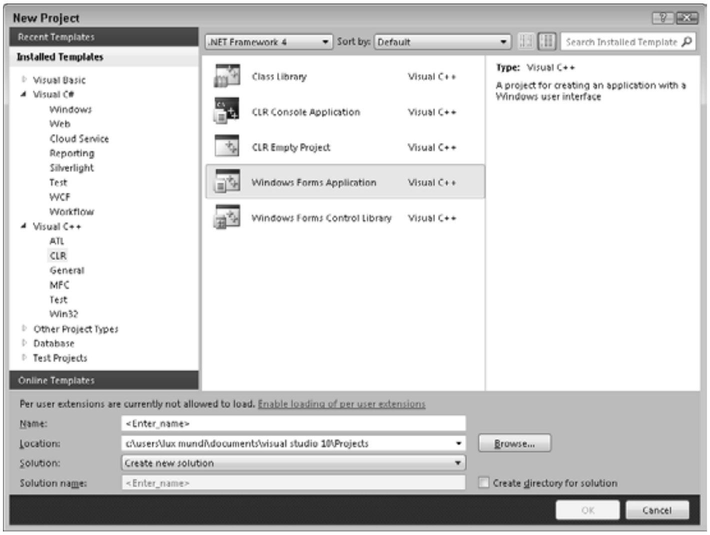
The Visual Studio project templates and application wizards make it easy to build managed and unmanaged Visual C++ applications. The steps for creating managed Visual C++ applications with project templates are virtually the same for building C# or Visual Basic applications. Whether you're using a project template or an application wizard, creating a Visual C++ project is similar to creating any other project in Visual Studio:

1. **Choose File⇨New⇨Project.**

   The New Project dialog box appears.

2. **In the Installed Templates tree, open the Visual C++ section to expand the list of available project types.**

   A list of available project types appears, as shown in Figure 3-1.



Type: Visual C++

A project for creating an application with a Windows user interface

Per user extensions are currently not allowed to load. Enable loading of per user extensions

Name: <Enter_name>

Location: c:\users\lux mundi\documents\visual studio 10\Projects

Solution: Create new solution

Solution name: <Enter_name>

□ Create directory for solution

Figure 3-1: Click a project type to see a list of available Visual C++ project templates.

Figure 3-1 shows the New Project window using the C++ development environment settings. You don't have to use the C++ development environment settings to create C++ projects. See Book II, Chapter 1 for more information on changing your development environment settings.

3. **Click the project type you want to create.**

   A list of available project templates for that project type appears in the Templates area of the screen.

4. **Click a project template.**
5. **Enter a name for your project in the Name text box.**
6. **Click OK to create your project.**

   For unmanaged applications, Visual Studio starts the appropriate wizard. For managed applications, Visual Studio creates the project and adds header, resource, and source files.

Creating managed applications

Creating a managed Windows Forms application in Visual Studio is the same as creating a Windows Forms application in any other .NET language. To create a new managed Windows application by using the Visual C++ language, follow these steps:

1. **Open the New Project window, as described in the preceding section.**
2. **In the Installed Templates tree, open the Visual C++ section.**
3. **Click the CLR project type.**

   A list of project templates available for creating a CLR application appears.

4. **Click the Windows Forms Application template.**
5. **Enter a name for the application.**
6. **Click OK to create the project.**

Visual Studio creates the project and opens a blank Windows Form in the Windows Forms Designer. The project created by Visual Studio includes these elements:

- **References:** Give your project access to the services of the .NET Framework.
- **Source files:** Jump-start your project.
- **Header files and resource files:** Support the project's source files.

To say "Hello world" in a managed Visual C++ Windows Form Application, follow these steps:

1. **Drag and drop a label control and a button control on the Windows Form.**
2. **Double-click the button control to access the control's `Click` event.**

   The Code Editor opens.
3. **Type this code in the Code Editor:**

   ```
   this->label1->Text = "Hello World";
   ```
4. **Press Ctrl+F5 to run your form.**

   When you click the button on the form, the text `Hello World` appears in the label.

Working with managed Windows Forms in the Windows Forms Designer is the same, regardless of the underlying programming language. Refer to Chapters 1 and 2 of Book IV for more information on using the Windows Forms Designer.

To say, "Hello World" in a managed Visual C++ Console Application, follow these steps:

1. **Add this code in the Code Editor:**

   ```
   Console::WriteLine(L"Hello World");
   ```

   The `L` constant preceding `"Hello World"` tells the compiler to treat this string as a wide character string (`wchar_t[]`) where each character contains two bytes of data. Conversely, in narrow character strings each character contains one byte.
2. **Press Ctrl+F5 to run your form.**

   When you click the button on the form, the text `Hello World` appears in the output.

Creating unmanaged applications

The Visual Studio application wizards help you step through the creation of unmanaged Visual C++ applications. After you complete an application wizard, use Visual Studio resource editors and code wizards to complete your project.

Using a wizard to create an unmanaged application

Visual Studio has several wizards for creating Visual C++ projects. Use the Visual Studio application wizard to

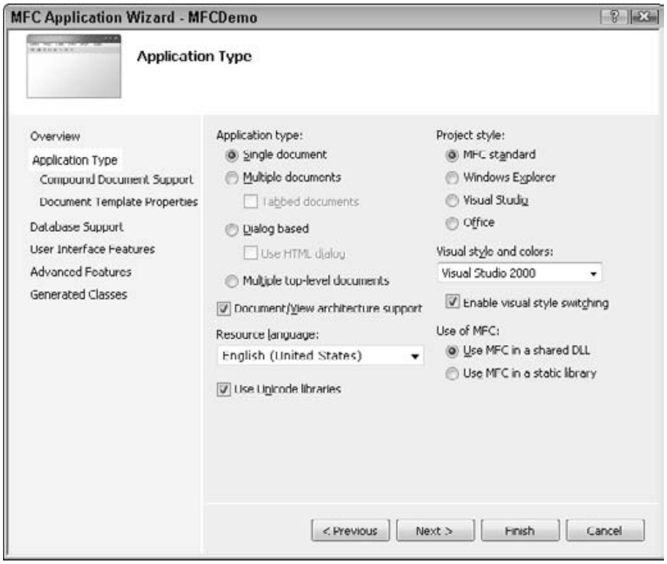
- Generate source code to create the program's basic structure
- Include resources, such as menus and toolbars
- Wire all the libraries to make the project work

To create a new Windows application by using MFC, follow these steps:

1. **Open the New Project window, as described at the beginning of this section.**
2. **Expand the list of Visual C++ project types and click MFC.**

   A list of available MFC application wizards appears.
3. **In the Templates pane, click the MFC Application icon.**
4. **Give your project a name and then click OK.**

   The MFC Application Wizard appears.
5. **Click Next to step through the wizard.**
6. **On the Application Type page, set the application type to Dialog, as shown in Figure 3-2.**



MFC Application Wizard - MFCDemo

Application Type

Application type:
● Single document
○ Multiple documents
□ Tabbed documents
○ Dialog based
□ Use HTML dialog
○ Multiple top-level documents
☑ Document/View architecture support

Resource language: English (United States)

☑ Use Unicode libraries

Project style:
● MFC standard
○ Windows Explorer
○ Visual Studio
○ Office

Visual style and colors: Visual Studio 2000

☑ Enable visual style switching

Use of MFC:
● Use MFC in a shared DLL
○ Use MFC in a static library

Figure 3-2: Set the application type to Single Document.

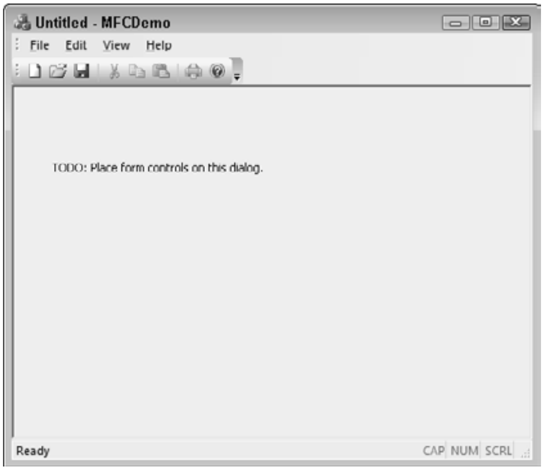
Use the Application Type page to specify the kind of Windows Form to create.

7. **Continue stepping through the wizard to set options for database support and user interface features.**

8. **Click Finish.**

   Visual Studio adds the source, header, and resource files that are necessary to build the options you specify in the wizard.

The wizard generates a fully functioning Windows application. Press Ctrl+F5 to run it. You should see a window similar to the one shown in Figure 3-3.



TODO: Place form controls on this dialog.

Figure 3-3: The application wizard generates a fully functioning Windows application.

Pick up a copy of *C++ For Dummies,* 6th Edition, by Stephen Randy Davis (Wiley), if you need to know about C++ syntax.

Managing resources

Visual Studio provides resource editors for working with a Visual C++ project's resources. Here are a few resource editors:

- **Dialog Editor:** Manages dialog boxes
- **Menu Editor:** Manages menus
- **String Editor:** Manages all your project's strings

To access a project's resources, follow these steps:

1. **Choose View➪Resource View or press Ctrl+Shift+E.**

   A tree view of the project's resources appears.

2. **Expand the resource folders until you see the Dialog folder.**
3. **Click the Dialog folder to expand it.**
4. **Double-click the Dialog resource `IDD_ABOUTBOX`.**

   The Menu Editor opens, as shown in Figure 3-4.

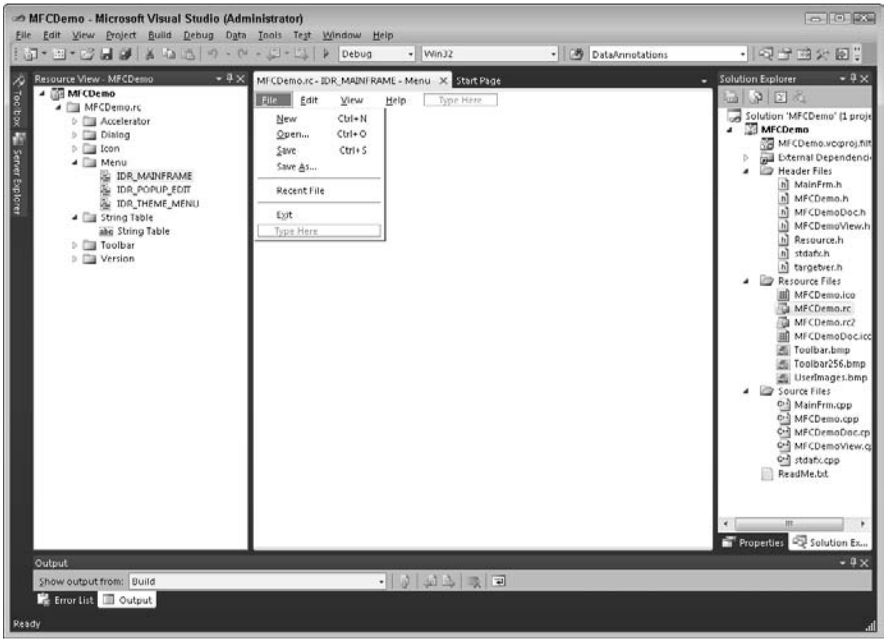


Figure 3-4: Use the Dialog Editor to edit a dialog box.

To add controls to your Windows Form, follow these steps:

1. **Click the plus sign (+) next to the Dialog folder in your project's Resource View pane.**

   A list of your project's forms appears.

2. **Double-click the form `IDD_TEST2_DIALOG` to open it.**

   The form opens in the Dialog Editor, as shown in Figure 3-5. Note that the name of the form is generated by using the project name you specify. If you used a project name other than MFCDemo, insert your project's name between IDD_ and _FORM.

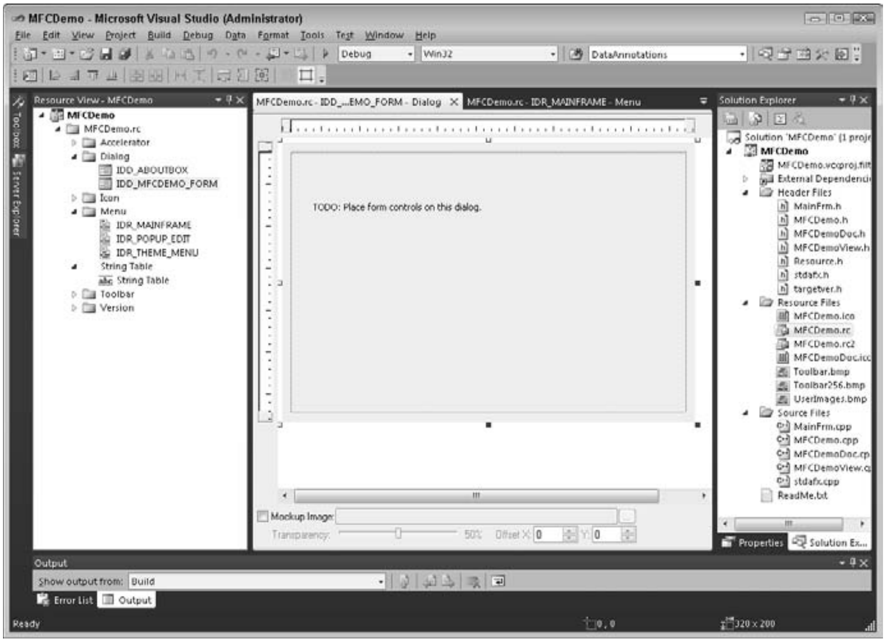


Figure 3-5: Use the Dialog Editor to add controls to your project's forms.

3. **Drag and drop controls from the toolbox onto the Dialog Editor.**

You can build event handlers for controls by choosing Add Event Handler from the control's shortcut menu.

To add a resource to a project, follow these steps:

1. **Click the Resource View pane.**
2. **Select the project in the tree view and choose Add➪Resource.**

   The Add Resource dialog box appears.
3. **Choose a resource from the list of resource types.**
4. **Click the New button, as shown in Figure 3-6, to add the resource.**

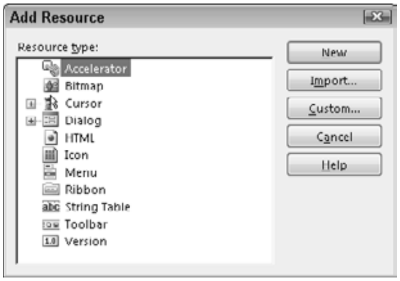


Figure 3-6: Use the Add Resource window to add resources to your project.

External dependencies

New in Visual Studio 2010 is a feature that scans your C++ project for external header files on which your application depends and lists them in the Solution Explorer, as shown in Figure 3-7.

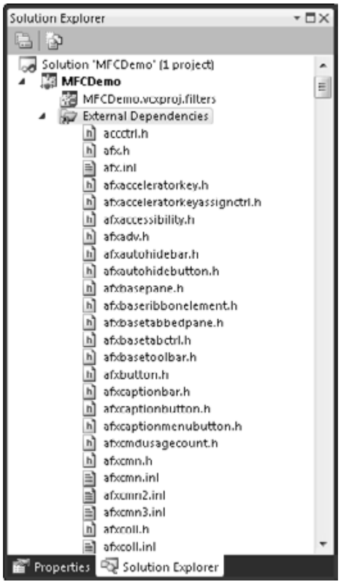


Figure 3-7: External Dependencies in Visual C++ Solution Explorer.

Visual Studio uses the external dependencies to run the parser on your C++ code against these header files to flag code that doesn't have correct syntax.

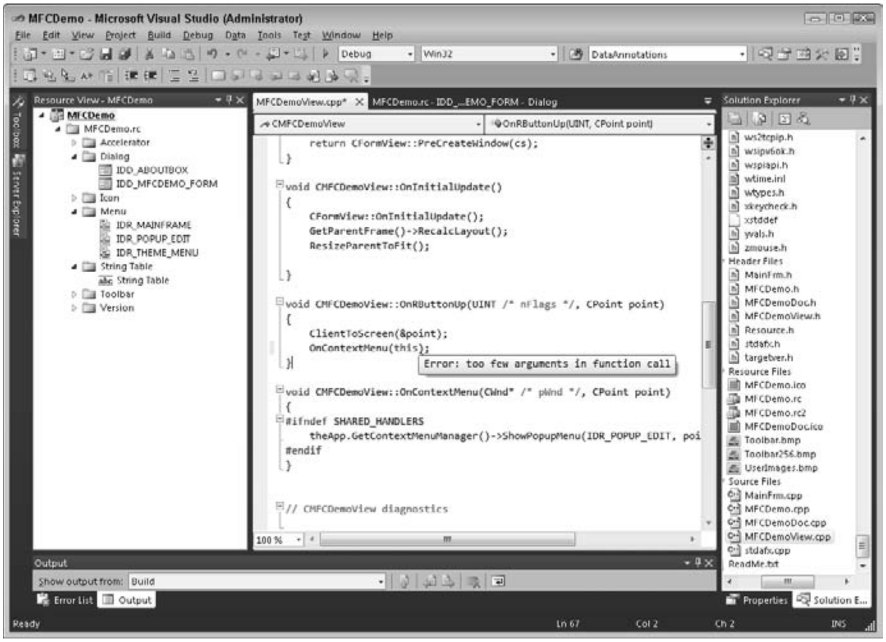
In `MFCDemoView.cpp`, if you remove the second argument to the `OnContextMenu` function in the `CMFCDemoView::OnRButtonUp` function, Visual Studio flags the syntax error with a tooltip, as shown in Figure 3-8.

Adding new external dependencies

In Visual Studio 2010, you can add external dependencies to your project as follows:

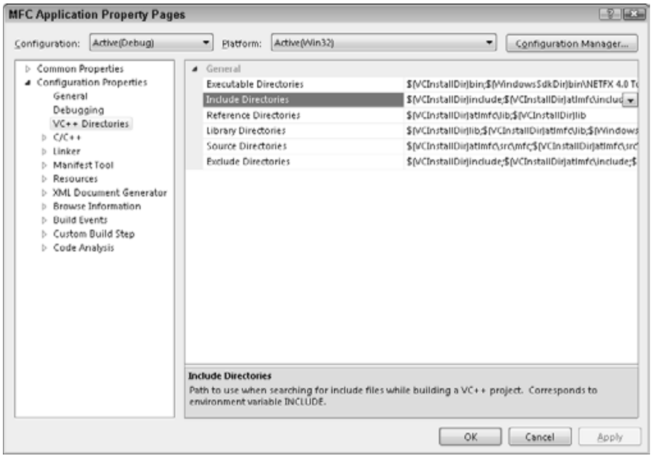
1. **Select the project name in the Solution Explorer.**
2. **Right click the project name and choose Properties from the contextual menu.**

   The Property Pages dialog box appears, as shown in Figure 3-9.



```
    return CFormView::PreCreateWindow(cs);
}

void CMFCDemoView::OnInitialUpdate()
{
    CFormView::OnInitialUpdate();
    GetParentFrame()->RecalcLayout();
    ResizeParentToFit();

}

void CMFCDemoView::OnRButtonUp(UINT /* nFlags */, CPoint point)
{
    ClientToScreen(&point);
    OnContextMenu(this);
}
                        Error: too few arguments in function call

void CMFCDemoView::OnContextMenu(CWnd* /* pWnd */, CPoint point)
{
#ifndef SHARED_HANDLERS
    theApp.GetContextMenuManager()->ShowPopupMenu(IDR_POPUP_EDIT, poi
#endif
}


// CMFCDemoView diagnostics
```

Figure 3-8: Automatic syntax checking using external dependencies.



Configuration: Active(Debug)   Platform: Active(Win32)

General	
Executable Directories	$(VCInstallDir)bin;$(WindowsSdkDir)bin\NETFX 4.0 To
Include Directories	$(VCInstallDir)include;$(VCInstallDir)atlmfc\includ
Reference Directories	$(VCInstallDir)atlmfc\lib;$(VCInstallDir)lib
Library Directories	$(VCInstallDir)lib;$(VCInstallDir)atlmfc\lib;$(Windows
Source Directories	$(VCInstallDir)atlmfc\src\mfc;$(VCInstallDir)atlmfc\src
Exclude Directories	$(VCInstallDir)include;$(VCInstallDir)atlmfc\include;$

Include Directories

Path to use when searching for include files while building a VC++ project. Corresponds to environment variable INCLUDE.

Figure 3-9: Adding new Include Directories to a C++ project.

3. **Select VC++ Directories on the left pane of the Property Pages dialog box.**

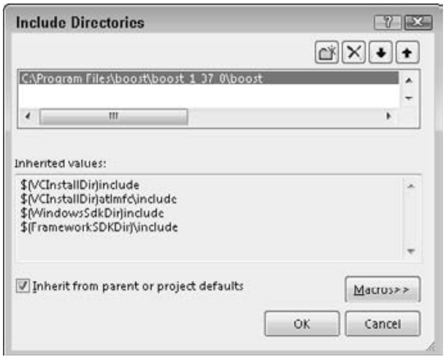
4. **In the right pane, select Include Directories and to the right of this pane click the down arrow and the <Edit...> from the window that appears.**

   The Include Directories dialog box appears, as shown in Figure 3-10.

5. **Browse to the directory that contains the required headers, as shown in Figure 3-11, and click OK.**

   The Include Directories dialog box closes, and the newly added directory is added to the Include Directories list in the right pane of the Property Pages dialog box (refer to Figure 3-9).

6. **Click OK to dismiss the Property Pages dialog box.**



```
C:\Program Files\boost\boost_1_37_0\boost
```

Inherited values:

```
$(VCInstallDir)include
$(VCInstallDir)atlmfc\include
$(WindowsSdkDir)include
$(FrameworkSDKDir)\include
```

☑ Inherit from parent or project defaults

Figure 3-10: Creating a reference to external header directories.

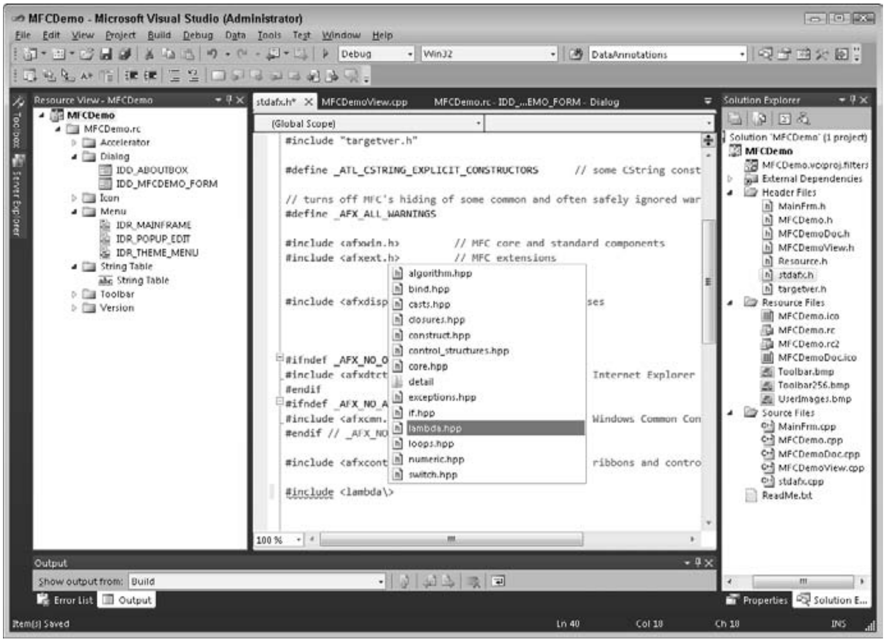
7. **Add a new `#include` directive in `stdafx.h`.**

   IntelliSense lists the header files in the referenced folder, as shown in Figure 3-12.

   The new header file `lambda.hpp` is added to the list of external dependencies in the Solution Explorer, as shown in Figure 3-12.

Linking with external libraries

If you're using an external library such as Boost, then you'll likely also need to link your code with a DLL provided by an external library. To tell the Visual C++ linker where this DLL is located, select the Executable Directories from the project Property Pages, as shown in Figure 3-9, and navigate to the folder in which the necessary DLL(s) are located just as you see in the section "Adding New External Dependencies." Now the Visual C++ linker is able to link this library with your code to create a new application.



```
#include "targetver.h"

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS      // some CString const

// turns off MFC's hiding of some common and often safely ignored war
#define _AFX_ALL_WARNINGS

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions


#include <afxdisp                                         ses



#ifndef _AFX_NO_O
#include <afxdtct                                         Internet Explorer
#endif
#ifndef _AFX_NO_A
#include <afxcmn.                                         Windows Common Con
#endif // _AFX_NO

#include <afxcont                                         ribbons and contro

#include <lambda\>
```

Figure 3-11: IntelliSense listing of external header files.

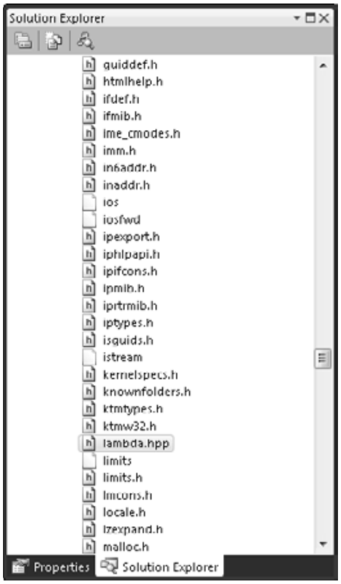


Figure 3-12: The new file is added to external dependencies in Solution Explorer.

Chapter 4: Building Web Applications

In This Chapter

- Using Visual Web Developer
- Creating your first Web site
- Understanding server controls
- Getting comfortable with ASP.NET syntax
- Creating user controls

This chapter focuses on Visual Web Developer, which is the Web development tool that comes with Visual Studio 2010. In this chapter, you see how to build Web applications using Visual Studio and the ASP.NET Web development platforms. You also discover the difference between server controls and HTML controls as well as how to add your own custom controls to your Web site.

Exploring Microsoft's Web Design and Development Tools

To help you build Web applications, Microsoft offers a wide array of products. Among them are ASP.NET and Visual Studio, which are used to create enterprise Web sites featuring e-commerce and full-blown Web-based applications. ASP.NET Web sites use server-side code to do many things, from rendering dynamic content to accessing databases.

In addition to these high-end tools, Microsoft offers Web design and development tools for a range of sites and skill levels, all of which support Web design standards, such as Cascading Style Sheets (CSS), to work together seamlessly. The bigger picture of Microsoft tools looks something like this:

- **Visual Web Developer:** The premiere Microsoft toolset for building ASP.NET Web sites is part of Visual Studio 2010.
- **Microsoft Expressions:** This set of three products (Graphic Designer, Interactive Designer, and Web Designer) targets professionals who design Web sites.

- **SharePoint Designer:** Because this tool for designing SharePoint sites is built on FrontPage (a program for creating HTML-based sites), SharePoint Designer can create the file-based sites of FrontPage. SharePoint Designer supports the limited development of ASP.NET Web sites.

The Microsoft Web design and development products recognize that building Web sites requires specialized skills. The person designing Web site graphics isn't usually the same person who's writing the site's data access procedures. Because the specialized tools "talk" to each other, Web designers, graphics designers, and developers can all work on different facets of the Web site without interfering with other areas or having to translate each other's work.

Getting to Know the Visual Web Developer

You can use Visual Web Developer for any of the following tasks:

- Create and edit Web sites by using ASP.NET.
- Design Web pages by using visual design tools.
- Test your Web sites on your choice of Web servers, including the lightweight ASP.NET Development Server.
- Use server-side ASP.NET controls for creating dynamic content.
- Publish your Web site to a host server.
- Use syntax-aware editors for HTML and server-side code.
- Access data by using server-based controls and wizards.
- Create a consistent "look and feel" for your Web sites by using themes and master pages.
- Build Web sites that have sophisticated features, such as personalization, membership, and site navigation.

The following sections help you get acquainted with Visual Web Developer.

Building better Web applications with ASP.NET

The ASP in ASP.NET stands for Active Server Pages. After people realized that using HTML to write Web pages by hand was a long and laborious chore, they figured out that they could replace static HTML content with server-side programs. For example, rather than write the same page header repeatedly, you can call a program on a server that magically spits out the page banner every time it's called. The ASP.NET code still generates HTML, but the developer doesn't have to write that common HTML in all the files.

Web Forms

ASP.NET Web pages are also called Web Forms because they use the HTML form element. The programming style for Web Forms is similar to Windows Forms, which may be another reason that the term Web Forms is used. Microsoft has mostly stopped using this term in favor of Web page. Interestingly, applications created to target the Aero interface in Windows Vista employ a user interface (UI) control called a page, not Windows Forms.

An ASP.NET Web site uses its own kind of Web pages. ASP.NET Web pages are different from plain old Web pages in the following ways:

- Files end in `.aspx` rather than the usual `.htm` or `.html` for Web pages.
- Directives in the form of

  ```
  <@ Page attribute="value">
  ```

  set configuration attributes used by the compiler.
- A form element in the following syntax

  ```
  <form id="formname" runat="server">
  ```

  is required on every ASP.NET Web page. The form control is responsible for intercepting a user's interaction with your Web page and sending it to the server for processing. But it does so differently than the standard HTML `<form>` tag that it uses and its code-behind to add much more functionality.

  Microsoft didn't invent the `form` tag. Rather, the `form` tag is part of the HTML syntax and is often called a server form. Many server-side scripting languages, such as the Practical Extraction and Report Language (PERL) and Hypertext Preprocessor (PHP), are used with the form tag to respond to actions taken by someone using an HTML form. Microsoft invented the `runat="server"` attribute, which ASP.NET uses to build the form so that it can be sent to the browser.
- You can use Web server controls to invoke server-side programs. I discuss these special elements in the section "Benefitting from the battle of the server controls" later in this chapter.

You use the visual design tools in Visual Web Developer to drag and drop Web server controls on an ASP.NET Web page. Visual Studio then generates the code to make the controls work, and you write your own custom code that processes the controls when users interact with them in their browsers. Visual Studio packages all this code and creates a compiled computer program that's executed whenever a user requests the page.

Understanding how ASP.NET pages are served

ASP.NET Web sites are hosted on the Microsoft Internet Information Services (IIS) Web server. The server intercepts all ASP.NET requests and passes the requests to ASP.NET for processing. The IIS Web server identifies an ASP.NET request from a plain old HTML request because ASP.NET Web pages use the `.aspx` file extension.

When you use Visual Studio to run an ASP.NET Web site, Visual Studio compiles the pages. When you deploy your Web site to an IIS Web server, the files are copied as-is, and ASP.NET takes over the process of compiling the code for you. You can force Visual Studio to compile the code before you deploy it to IIS. See Book VI, Chapter 2 for more information about deploying ASP.NET Web sites.

Any time a user requests an ASP.NET Web page in her browser, a program runs on the Web server to deliver the page. When the user performs an action on the page, such as clicking a button to submit information, the page sends a request back to the server. The process of sending data back to the server is a *postback.*

The postback calls the same ASP.NET Web page that the user interacted with. The data the user entered into the browser is passed to the ASP.NET Web page's server-side code for processing. The server-side code customarily sends back a confirmation message to the user indicating whether the data was successfully processed.

A postback is also referred to as a *round trip* because data flows from the browser to the server and back to the browser each time the user performs an action on the page. For more information about writing the server-side code that responds to a postback in an ASP.NET Web page, see Book V, Chapter 4.

You can avoid postbacks in many cases. You can use validation controls, which inject script into the HTML to perform the validation tasks without a postback. AJAX adds even more functionality on the client side, thus avoiding even more postbacks.

Creating Web sites

You can use the Visual Web Developer in Visual Studio to create different kinds of Web sites, depending on where you want to work with the site's content and the kind of Web server you want to use for testing. Here are the four kinds of sites you can create with Visual Studio:

- **File system:** Store your Web site's files in a folder on your local hard drive and use the Visual Studio built-in Web server for testing.
- **Local IIS:** Create a Web site on your local machine by using IIS as your Web server. Visual Studio creates the Web site in IIS for you.

- **Remote server:** Access an IIS Web server on a different computer by using the HTTP protocol.
- **FTP:** Access a Web server on a different computer by using the File Transfer Protocol (FTP). You typically use FTP when you access a site hosted on a third-party server.

Note that the kind of Web site you create doesn't determine how you choose to deploy it. That is, you can create a file system Web site and then later deploy it to an IIS server by using FTP. For more information on deployment options, see Book VI, Chapter 2.

Working with a file system Web site is generally much easier than working with any of the other configurations. If more than one developer is working on a single Web site, you might need to use one of the other approaches. For more information about team development and source code control, see Book VI, Chapter 3.

Saying "Hello, World Wide Web"

Creating a Web site in Visual Studio is the easy part of building a Web application. Choosing the content that goes into your Web application is a little trickier. When you create a Web site, Visual Studio does all the work for you. Visual Studio creates the folder structure and configuration files necessary to run the Web site. Here's what ASP.NET does when it creates a Web site:

- Creates a folder structure on your local computer for managing the Web site's files
- Provisions the project to the specified Web server
- Creates a new Web Forms page named `default.aspx`
- Creates a code-behind file (the code that provides the functionality to the presentation layer) for writing server-side code for the forms page named `default.aspx.vb`
- Adds an `App_Data` folder for storing data-related files
- Creates a `web.config` file for storing Web site configuration information

This book describes folder-based systems exclusively because they're easy to work with.

To create a new folder-based Web site, follow these steps:

1. **Choose File⇨New Web Site.**

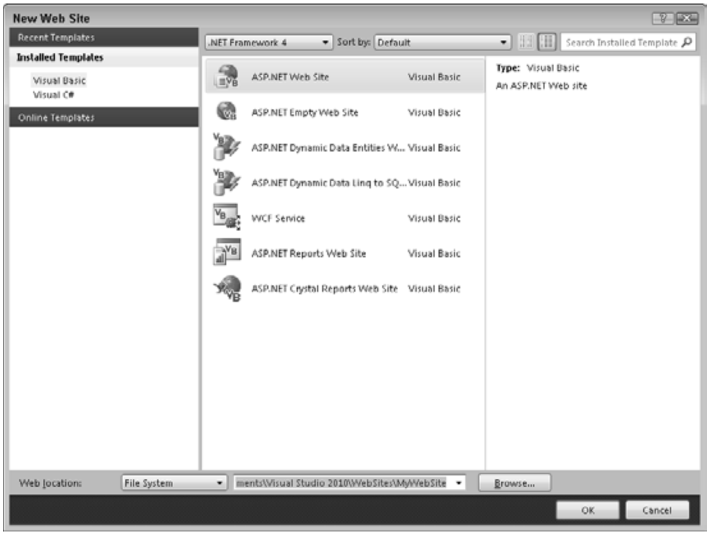
   The New Web Site dialog box appears.

2. **Click the ASP.NET Web Site template.**

   The default location for a new Web site is the file system.

3. **Accept the default location and enter a name, such as MyWebSite, after the pathname, as shown in Figure 4-1.**

   Leave the language set to Visual Basic.



New Web Site

.NET Framework 4

Sort by: Default

Type: Visual Basic

An ASP.NET Web site

Web location: File System ments\Visual Studio 2010\WebSites\MyWebSite

Recent Templates
Installed Templates
Visual Basic
Visual C#
Online Templates
Search Installed Template
ASP.NET Web Site Visual Basic
ASP.NET Empty Web Site Visual Basic
ASP.NET Dynamic Data Entities W... Visual Basic
ASP.NET Dynamic Data Linq to SQ... Visual Basic
WCF Service Visual Basic
ASP.NET Reports Web Site Visual Basic
ASP.NET Crystal Reports Web Site Visual Basic
Browse...
OK
Cancel

Figure 4-1: Append the name of the Web site to the file path.

4. **Click OK to create the ASP.NET Web site.**

   The default.aspx page opens in Design view.

Goodbye, IIS; hello, ASP.NET Development Server

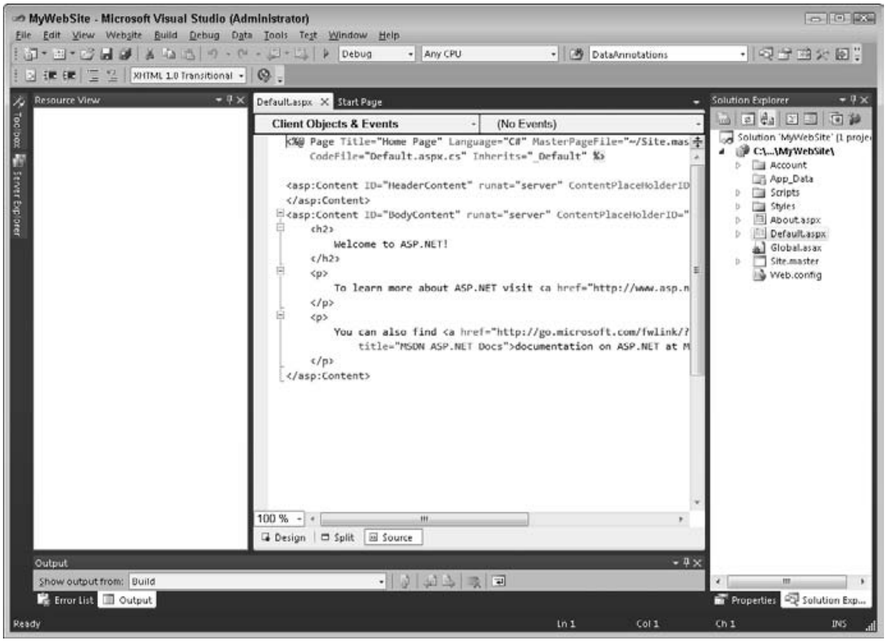
Because early versions of Visual Studio didn't include a Web server, most developers used a local copy of IIS installed on their computers. Although IIS is no longer required for testing ASP.NET Web sites, it's still required for running ASP.NET Web sites in a production environment.

IIS adds a layer of complexity that you may not want to deal with. However, because IIS is the production Web server, you need to test your site with it, anyway.

Because file-based Web sites use the ASP.NET Development Server, you don't need to do any configuration.

Viewing ASP.NET syntax

An ASP.NET Web page is a combination of HTML markup elements and ASP.NET syntax. The `runat="server"` attribute is an example of ASP.NET syntax that tells the ASP.NET compiler to process tags differently. Any plain-text or HTML tags are rendered to the browser exactly as they appear on the Web page. You use the Source view of an ASP.NET page to view its syntax. To view the `default.aspx` page in Source view, click the Source tab at the bottom of the Document Explorer, as shown in Figure 4-2.



```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.mas
    CodeFile="Default.aspx.cs" Inherits="_Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="
    <h2>
        Welcome to ASP.NET!
    </h2>
    <p>
        To learn more about ASP.NET visit <a href="http://www.asp.n
    </p>
    <p>
        You can also find <a href="http://go.microsoft.com/fwlink/?
            title="MSDN ASP.NET Docs">documentation on ASP.NET at M
    </p>
</asp:Content>
```

Figure 4-2: Click the Source tab to display the page in Source view.

Notice the markup elements displayed in Source view:

- The `<%@ Page . . . %>` directive at the top of the page provides additional configuration information for the ASP.NET compiler.
- The `<!DOCTYPE . . . >` entry tells the client's Internet browser that the page complies with the XHTML standard.

  If you don't care about this standard, you can safely delete this line.
- Familiar HTML tags, such as `<html>` and `<head>`, are used.
- The `runat="server"` attribute is used on the `<head>` and `<form>` tags.

The syntax you see displayed in Figure 4-2 appears in every Web page you add to your project. You drag and drop controls from the toolbox onto the

Web page's Design view, as described in the next section. As you add controls to the page, ASP.NET syntax appears in the page's Source view.

You can use the View menu to access the Web page's Source and Design views.

Adding content

The Web page that's added to a newly created ASP.NET Web site is blank. To make the page meaningful, you must fill it with content, such as

- Text boxes, labels, and buttons
- Plain text in paragraphs or lists
- Images and animations
- Tables displaying data and forms that allow users to enter and interact with data

You have several options for adding content to your Web page. If you know the ASP.NET syntax for building ASP.NET Web pages, you can just open any text editor and start adding content. But most people prefer to use Design view in Visual Studio to add content so that they can see how the content they're adding appears on the page as they go.

Using the Visual Designer to add content in Visual Studio 2010 involves these three steps:

1. **Drag and drop the controls on the visual design surface.**
2. **Set the properties that determine attributes, such as appearance and function.**
3. **Specify what happens (if anything) when a user interacts with a control by either using a wizard to build the code or writing the code manually.**

Suppose that you want to create a simple Web form page that displays the text `Hello World!` when a user clicks a button. To add the content to the default.aspx Web page that you can see how to create in the preceding section, follow these steps:

1. **In Design view of the default.aspx page, drag and drop a Label control from the Toolbox task pane.**

   If the toolbox isn't visible, press Ctrl+Alt+X.

2. **Drag and drop a Button control on the designer surface.**

   A button appears on the page.

3. **Press Ctrl+F5 to run the Web page.**

The Web page that appears has a label and a button. If you click the button, however, it doesn't do anything. To display `Hello World!`, you have to finish setting the controls' properties and configuring the button to work.

ASP.NET projects that use the File System for deployment execute using the ASP.NET Development Server. When you run a Web page from Visual Studio, the ASP.NET Development Server executes and loads the Web page in your browser. A text balloon appears in the Notification Area displaying the URL of your Web page. The ASP.NET Development Server also displays an icon in the Notification Area. You can stop this service by right-clicking this icon and choosing Stop from the contextual menu.

To set the controls' properties, follow these steps:

1. **In Design view, press F4 to open the Properties window.**
2. **Click the Label control in the designer to display the label's properties in the Properties window.**

   The Properties window lists properties on the left and their corresponding values on the right.
3. **Locate the Text property and then set the value to Blank by deleting the contents.**
4. **Locate the ID property and name it `lblMessage`.**

Repeat these steps to set the button's `Text` property to `Show Message`, and its ID to `btnDoIt`. The `Text` property specifies the text that appears on the page.

Adding the code to make the button work requires you to work in Code view on the page. Although you can type the code directly in the Code Editor, it's easier to start in Design view and let Visual Studio write some of the hookup code for you. Follow these steps:

1. **In Design view, double-click the button you placed on the designer.**

   The Code Editor appears. Double-clicking any control in the designer creates a block of code that's linked to the default event handler for that control. In the case of a button, the default event handler is the `Click` event. The code that's typed in the code block is executed every time the button is clicked. Visual Studio automatically places the insertion point in the `btnDoIt_Click()` method. For more information on event handlers, see Book III, Chapter 4.
2. **In the Code Editor, type this line:**

   - **VB**

     ```
     lblMessage.Text = "Hello World!"
     ```

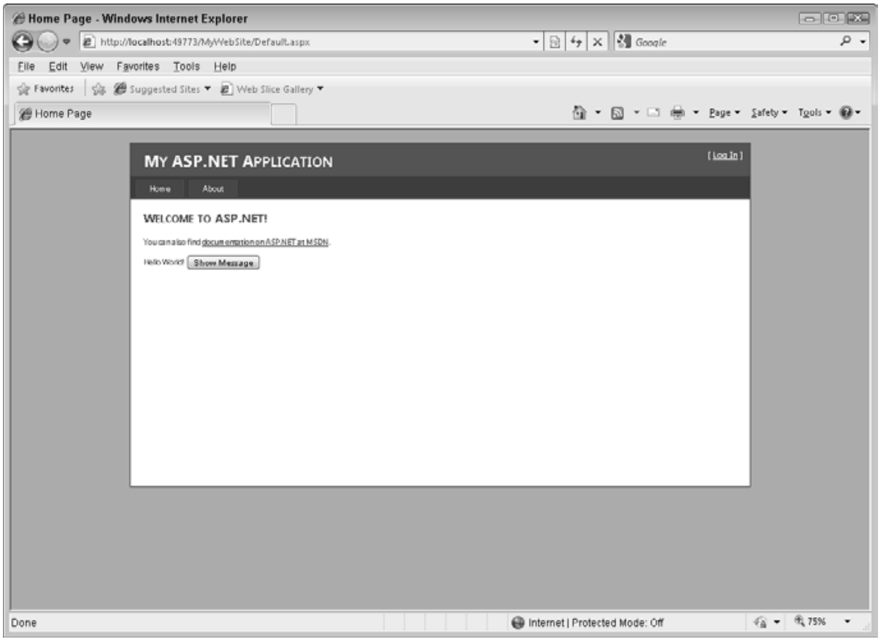
   - **C#**

     ```
     lblMessage.Text = "Hello World!";
     ```

3. **Press Ctrl+F5 to run the Web page.**

4. **When the Web page appears, click the button.**

   The statement `Hello World!` appears on the page, as shown in Figure 4-3.


Home Page - Windows Internet Explorer
http://localhost:49773/MyWebSite/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page Safety Tools
MY ASP.NET APPLICATION
[Log In]
Home
About
Show Message
Done
Internet | Protected Mode: Off
75%

WELCOME TO ASP.NET!

You can also find documentation on ASP.NET at MSDN.

Hello World!

Figure 4-3: The ASP.NET Web page displays this message when you click the button.

You can download the complete program (both C# and VB versions) for this example from the companion Web site for this book at `www.dummies.com/go/vs2010`.

Some of you may be thinking that something mysterious happens with ASP.NET applications. If you choose View Source from the browser's contextual menu, though, you'll see that the output of the ASP.NET application is common HTML. Yes, you write your code in a combination of HTML and C# or VB, but the output is simple HTML.

Working with Web Sites

As described in the "Creating Web Sites" section in this chapter, creating a Web site using Visual Studio is easy. The hard part is figuring out what content to add. Luckily for you, Visual Studio lets you add all kinds of content to your Web pages and Web site, and you have a ton of different ways to view and edit your Web pages.

Adding new pages

Adding new pages involves more than just adding Web pages. You can use Visual Studio to add these elements:

- **Web pages:** ASP.NET Web forms, Web services, mobile Web forms, user controls, and plain old HTML files, for example
- **Web pages for mobile Web sites:** Mobile Web forms and user controls, for example
- **Files to control the look and feel of your Web site:** Master pages, site maps, skin files, and CSS, for example
- **Files for managing data:** XML files, class files, and datasets, for example

Use the Add New Item dialog box to add new content to your Web site. You can open this dialog box in two ways:

- Right-click the project folder in Solution Explorer and choose Add New Item.
- Choose Web Site➪Add New Item.

You can add items that you already created by choosing the Add Existing Items option. Many of these items, such as the datasets, are displayed in their own designers. You don't have to write everything from scratch.

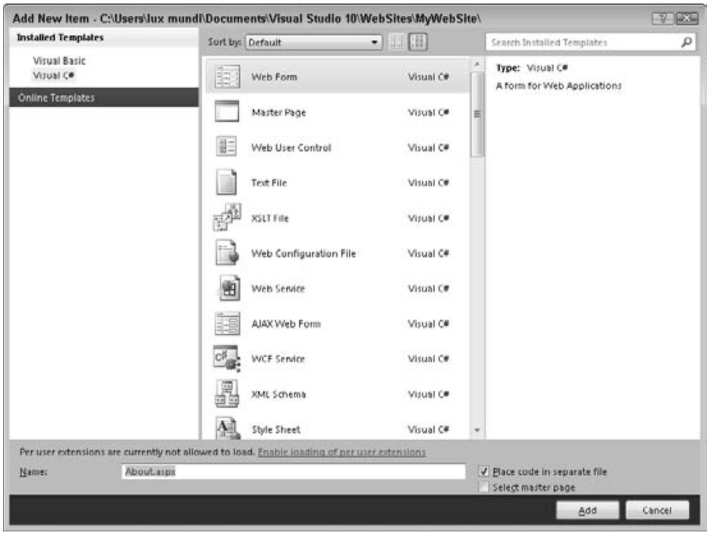
To add a new Web page to an existing Web site, follow these steps:

1. **Choose Website➪Add New Item.**

   The Add New Item dialog box, shown in Figure 4-4, appears.
2. **Click the Web Form template.**
3. **Enter the name `About.aspx` and leave the language as Visual Basic.**
4. **Leave the Place Code in Separate File check box selected and the Select Master Page check box deselected.**

   The Place Code in Separate File option places all your program code in a separate, code-behind file. Master pages are used to control the look and feel of your Web pages. See Book IV, Chapter 6 for more information about using master pages.
5. **Click Add.**

A new ASP.NET Web page is added to your Web site. The page opens in Design view so that you can add content. See Book IV, Chapter 6 for more information on site navigation and linking to Web pages.



Add New Item - C:\Users\lux mundi\Documents\Visual Studio 10\WebSites\MyWebSite\

Web Form	Visual C#
Master Page	Visual C#
Web User Control	Visual C#
Text File	Visual C#
XSLT File	Visual C#
Web Configuration File	Visual C#
Web Service	Visual C#
AJAX Web Form	Visual C#
WCF Service	Visual C#
XML Schema	Visual C#
Style Sheet	Visual C#

Type: Visual C#

A form for Web Applications

Per user extensions are currently not allowed to load. Enable loading of per user extensions

Name: About.aspx

☑ Place code in separate file

□ Select master page

Figure 4-4: Add a new Web page to your Web site.

You can add empty folders for organizing your content. ASP.NET Web sites use several reserved folders, which you can add to your site at any time:

- **`App_Browsers`:** Stores browser definition files, which are used to tell ASP.NET how to render markup for certain kinds of browsers, such as on mobile devices
- **`App_Code`:** Contains source code for class libraries, helper utilities, and business objects
- **`App_Data`:** Stores data files, including a local database for managing Web site membership and roles
- **`App_GlobalResources` and `App_LocalResources`:** Contains resource files used for translating page content into other languages
- **`Theme`:** Stores all your skins, CSS, images, and other “look and feel” files
- **`App_WebReferences`:** Contains files for Web services
- **`Bin`:** Stores compiled assemblies that you want to use in your Web site

You can add any of these folders by either choosing Website➪Add ASP.NET Folder or right-clicking the project folder and then choosing Add ASP.NET Folder. You can also right-click any of these folders in Solution Explorer and then choose Add New Item to display an abbreviated list of items you can add in that folder.

Benefitting from the battle of the server controls

ASP.NET provides several different kinds of controls you can use on your Web pages. You use controls to display dynamic content on your Web page.

ASP.NET writes all the plumbing code that renders your content and passes a user's response back to the server for you to process.

ASP.NET provides these kinds of server controls:

- **Web server:** Simple controls (such as a label or text box) and complex controls (such as calendars and menus)
- **HTML server:** Simple HTML elements that you expose for server-side coding
- **Validation:** A special kind of Web server control that validates the input of other Web server controls
- **User:** A custom control you create by using ASP.NET markup elements that can be embedded in a Web page

Server controls have many similarities. All of them

- Have the `runat="server"` attribute in their markup code.
- Must be placed inside an ASP.NET Web page in order to execute.
- Are accessible from server-side code, which means that you can write a program to manipulate the controls.
- Have events.
- Have properties that set their behavior at run-time.
- Must reside within the form control. (Each page can have only one.)
- Are rendered on the browser as a combination of plain HTML and JavaScript.

You can access all these server controls, except user controls, from the toolbox. You add user controls by using Solution Explorer. You can place other kinds of controls in a user control to create a composite control.

The syntax

The ASP.NET syntax for Web server controls looks similar to HTML markup syntax. Like with HTML, ASP.NET syntax uses tags with attribute/value pairs. The markup for a Web server button control looks like this:

```
<asp:Button ID="Button1" runat="server" Text="Button" />
```

You can add attributes to the markup code by either typing them directly in Source view of the Web page or by using the Properties window. The attributes specified in the markup code are properties of the Web server control, not HTML attributes. Some properties are mapped to HTML attributes. ASP.NET uses the collection of properties to determine how best to render the markup in a browser.

The markup code for an HTML server control looks like this:

```
<input id="Button2" type="button" value="button" runat="server" />
```

All server-side controls have these markup elements:

- **`ID` attribute:** Tells the server-side code how to access the control
- **`Runat` attribute:** Flags the control as a server-side control

Notice that the tags are a little different for Web server controls and HTML server controls. Web server control tags start with `asp:` and then the name of the control — for example, `asp:button` and `asp:textbox`. HTML server controls use HTML syntax, such as `<input>` and `<div>`. You can convert any HTML element to a server control by simply adding the `runat` attribute to the markup.

Server controls provided by ASP.NET use the tag prefix `asp`. Custom server controls created by you or a third party use the tag prefix defined for the custom server control. The tag prefix maps the control to the control's namespace. The namespace tells ASP.NET where to find the code that defines how the control works. Tag prefixes are defined by using a Register directive at the top of each Web page that uses the control. Alternatively, you can place the prefix information in the Web site's configuration file so that all pages in the site can use the custom server control.

All server controls must be contained within the `<form>` element on the Web page.

Web server controls versus HTML server controls

At first blush, HTML server controls and Web server controls look similar. They have some differences, however, as Table 4-1 illustrates.

Table 4-1 Comparison of HTML Server Controls and Web Server Controls

HTML Server Controls	*Web Server Controls*
Map almost one-to-one to HTML elements	Render to a single HTML element or a combination of elements
Are client-side by default	Server-side only
Properties mapped to HTML attributes	Properties aren't mapped directly to HTML attributes
Customized using CSS	Customized using CSS or templates
Don't support themes	Support themes

Here are some examples of Web server controls:

- `Label`, `TextBox`, and `Button`
- `LinkButton`, `ImageButton`, and `HyperLink`
- `TreeView` and `SiteMapPath`

To add an HTML control to a Web page and convert it to a server control, follow these steps:

1. **Click the plus sign (+) next to the HTML tab in the toolbox to expand the list of HTML controls.**

   If the toolbox isn't displayed, press Ctrl+Alt+X to open it.

2. **Drag an `Input (Button)` control to the design surface and drop it.**

   The image of a button appears.

3. **Edit the markup code with the `runat` property set to `server"` as follows:**

   ```
   <input id="Button1" type="button" value="button" runat="server"/></p>
   ```

All the events captured for this control are now server-side. Double-click the button's image to display a server-side code block, where you can enter server-side code for the button's `Click` event.

To convert the HTML control back to a regular control, you can remove the `runat` property from the control's markup in Source view.

REMEMBER

Unless you plan to use server-side resources, you should always use plain HTML elements. Server-side controls use more resources than client-side controls. You should also consider whether you can deliver functionality in a client-side script rather than use server-side code. By using a client-side script, you eliminate a round trip back to the server. (For a little background on round trips and postbacks, see the earlier section "Understanding how ASP.NET pages are served.")

For example, you can easily create a `Hello World!` example by using a client-side script. Follow these steps:

1. **Add a paragraph element to your Web page by typing the following code in Source view on the Web page:**

   ```
   <p id="output" style="color: Red; font-weight: bold"></p>
   ```

2. **Drag and drop an Input (Button) control in Design view on the page by using the HTML tab in the toolbox.**

3. **Enter** Show Message **in the `Value` property on the Properties window.**

   If the Properties window isn't displayed, press F4.

4. **Double-click the image of the Input button.**

   A JavaScript code block appears in Source view on the page. (JavaScript is a client-side scripting language.)

5. **Type this code in the JavaScript code block:**

   ```
   document.getElementById("output").innerHTML = "Hello World!";
   ```

6. **Press Ctrl+F5 to run the Web site.**

   The Web site appears in your browser.

7. **Press the button.**

   The phrase `Hello World!` appears on the page replacing the button.

You can find a link to a simple page demonstrating the JavaScript for the preceding example at the companion Web site for this book at `www.dummies.com/go/vs2010`.

This code sample uses the `innerHTML` property of the paragraph element. (This property was used for brevity.) The appropriate way to modify page content dynamically is to access the node and modify it by using the Document Object Model (DOM). The benefit of using `innerHTML` is that you can easily understand what this code is doing. The downside is that because the property isn't universal and doesn't provide a reference to the node in question, accessing the node again by using client-side code is impossible.

In the earlier "Adding Content" section, this chapter uses a Web server button control. How do you know when to choose an HTML input button and when to choose a Web server button control? Here are a couple of reasons to choose HTML input button controls instead of Web server button controls:

- **HTML server controls provide more control over the HTML markup that's rendered in the browser.** You can also control the markup of Web server controls by using a template, but it requires more work than using HTML server controls.
- **HTML server controls are easier for non-Visual Studio users to understand.** This information is especially valuable if you're working with Web designers who provide you with CSS because the designers may not understand how to provide styles for Web server controls.

On the other hand, here are some reasons to choose Web server button controls instead of HTML input button controls:

- **You want access to more properties than are available with a typical HTML element.**

- **You don’t want to take the extra step of converting the controls to server-side, like you do with HTML server controls.** By default, Web server controls are server-side.
- **You want to take advantage of device filters and browser definition files.**

Visual Studio provides more visual design support for Web server controls than for HTML server controls. If you want to rely on HTML server controls, why use ASP.NET? Many Web developers use Web server controls almost exclusively, except for the two generic HTML tags `div` and `span`.

You can also use a Label Web server control to display messages. Label controls are rendered as HTML `span` tags. The `span` tag is a generic inline container for use inside block-level containers, such as `div` tags and paragraph `<p>` tags. Because messages are usually rendered at the block level, some folks prefer to use `div` tags. If you plan to display a message inside a block tag, such as a paragraph tag, you can choose a `Label` control. In reality, most browsers aren’t affected if your `div` and `span` tags are improperly nested. It matters only if you’re a purist.

For an in-depth discussion of additional Web server controls, see Book IV, Chapter 6. It describes site navigation controls and hyperlinking mechanisms for transitioning between Web pages.

I cover login controls in Book IV, Chapter 6 and data controls in Book V, Chapter 2.

User controls and custom controls

Sometimes, you want to create a reusable control that doesn’t already exist in the Visual Studio toolbox. You can choose from two types:

- **User:** Create this type of control with the same tools you use to build Web pages. Like other controls, user controls are contained within an ASP.NET Web page.
- **Custom:** Build your own, custom control and add it to the Visual Studio toolbox. Writing a custom control requires some programming expertise.

User controls are similar to ASP.NET Web pages because you can include HTML and server-side code in a single file. Unlike Web pages, user controls

- End in the file extension `.ascx`
- Use the `@ Control` directive, rather than `@ Page`
- Must be hosted as part of an ASP.NET Web page
- Don’t include host page elements, such as `html`, `body`, and `form`

In previous versions of Visual Studio, user controls were used extensively as a means of providing page templates. For example, you would create header and footer user controls on each of your Web pages. That's no longer necessary with master pages.

This doesn't imply that user controls are no longer valuable. You can use a user control any time you want to encapsulate a group of controls and make them reusable. For example, you could create a user control for displaying calendar controls.

To create this user control in an existing Web site, follow these steps:

1. **Right-click the Web site's project folder in Solution Explorer and then choose Add New Item from the contextual menu.**

   The Add New Item dialog box appears.

2. **Click the Web User Control icon.**
3. **Give the user control the name `calendar.ascx` and then choose Visual Basic as the language and click Add.**
4. **Drag the control from Solution Explorer onto your Web Form.**

   Visual Studio opens the user control in Design view.

To add your user control to an existing Web page, drag the `calendar.ascx` file from Solution Explorer and drop it on Design view of the Web page. Visual Studio adds the following markup to your Web page:

- At the top of the page, you see

  ```
  <%@ Register src="calendar.ascx" TagName="calendar" TagPrefix="uc1" %>
  ```

- In the page content, you see

  ```
  <uc1:calendar ID="Calendar1" runat="server" />
  ```

This example shows a tag prefix being used for a custom control, as described in the earlier section "The syntax." The `Register` directive specifies the name of the control (`calendar`, in this case) and the tag prefix. To use the control on the page, the syntax references the custom control's tag prefix, `uc1`, and places the control's name after the colon. In this example, the control isn't mapped to a namespace. Instead, the control is mapped to the user control file using the attribute `Src="calendar.ascx"`. That's how ASP.NET knows where to find the control when it processes the control on the page.

In Design view of the Web page, the user control is displayed with a placeholder box until content is added to the user control. To open the user control for editing, follow these steps:

1. **In Design view of the Web page, click the smart tag arrow on the user control.**

   The UserControl Tasks list appears.

2. **Click the Edit UserControl link, as shown in Figure 4-5.**

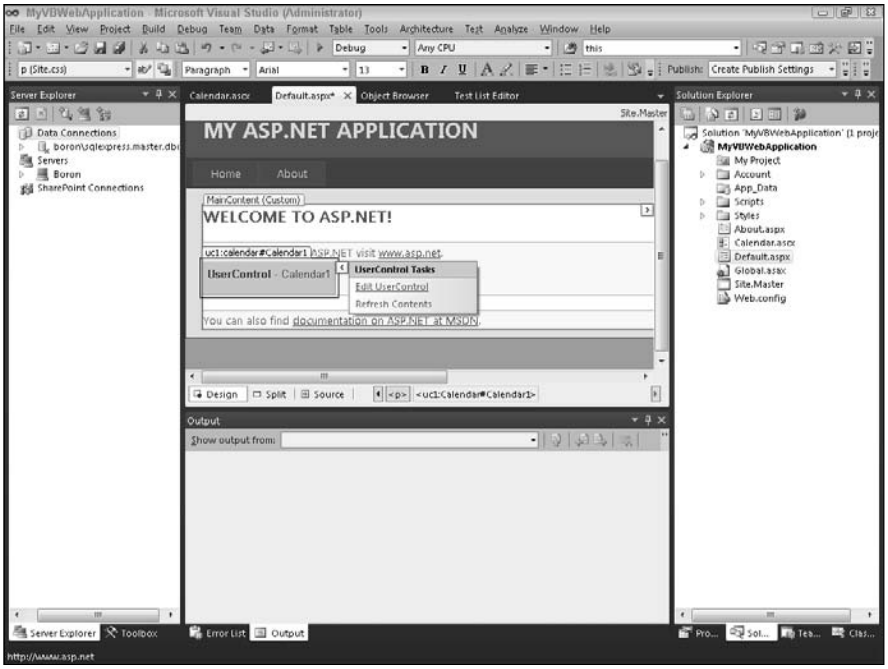


Figure 4-5: Click Edit UserControl to open the control for editing.

The user control opens for editing.

A user control is blank when you first create it. You add controls to newly created controls. Adding controls to a user control is similar to adding controls to a Web page. You simply drag and drop controls from the toolbox onto the design surface.

Because user controls are made up of other controls, they're often called *composite controls*.

One control that you can't use on a user control is the `<form>` server control. User controls, like all server controls, are hosted in Web forms. You can't call a user control as a normal HTML element. Web forms can have only one `<form>` server control. You place your user control inside the `<form>` server control on the page where you want to display your user control.

If you happen to place a `<form>` server control in a user control, you don't receive an error message until you execute the page where the user control is hosted.

The following example demonstrates how to create a user control — specifically, a specialized calendar control that displays the preceding, current, and following months. This user control encapsulates the controls and code that make the calendars work so that you can reuse the control without having to customize the code each time.

Follow these steps to create the calendar user control:

1. **Drag and drop a calendar control from the toolbox to Design view of the user control.**

   You can press Ctrl+Alt+X to display the toolbox.

2. **Add two more calendar controls by repeating Step 1 or copying and pasting the markup code in Source view.**

   You should have three calendar controls: `Calendar1`, `Calendar2`, and `Calendar3`.

3. **Position your controls so that `Calendar2` is between `Calendar1` and `Calendar3`.**

4. **On `Calendar1` and `Calendar3`, use the Properties window to set the `ShowNextPrevMonth` property to `false`. Set the `SelectionMode` property to `None`.**

   Press F4 to open the Properties window if it's closed.

5. **With the user control displayed in Design view, double-click a blank area of the designer.**

   Visual Studio opens the Code Editor and positions your cursor inside a code block for the `Page_Load` event.

   The `Page_Load` event is called when the user control is loaded.

6. **In the `Page_Load` code block, enter this code:**

   - **VB**

     ```
     If (Not Page.IsPostBack = True) Then
        Me.Calendar2.VisibleDate = System.DateTime.Today
        Me.Calendar1.VisibleDate = Me.Calendar1.TodaysDate.AddMonths(-1)
        Me.Calendar3.VisibleDate = Me.Calendar1.TodaysDate.AddMonths(1)
     End If
     ```

   - **C#**

     ```
     if( !IsPostBack )
     {
        Calendar2.VisibleDate = System.DateTime.Today;
        Calendar1.VisibleDate = Calendar1.TodaysDate.AddMonths(-1);
        Calendar3.VisibleDate = Calendar1.TodaysDate.AddMonths(1);
     }
     ```

TIP

If you want to test whether your code is working, press Ctrl+F5 to run your Web site.

7. **In Design view of the user control, click `Calendar2` and then press F4 to display the Properties window for that control.**

8. **Click the Events button on the Properties window toolbar.**

   The Events button looks like a lightning bolt.

9. **Double-click the `VisibleMonthChanged` event.**

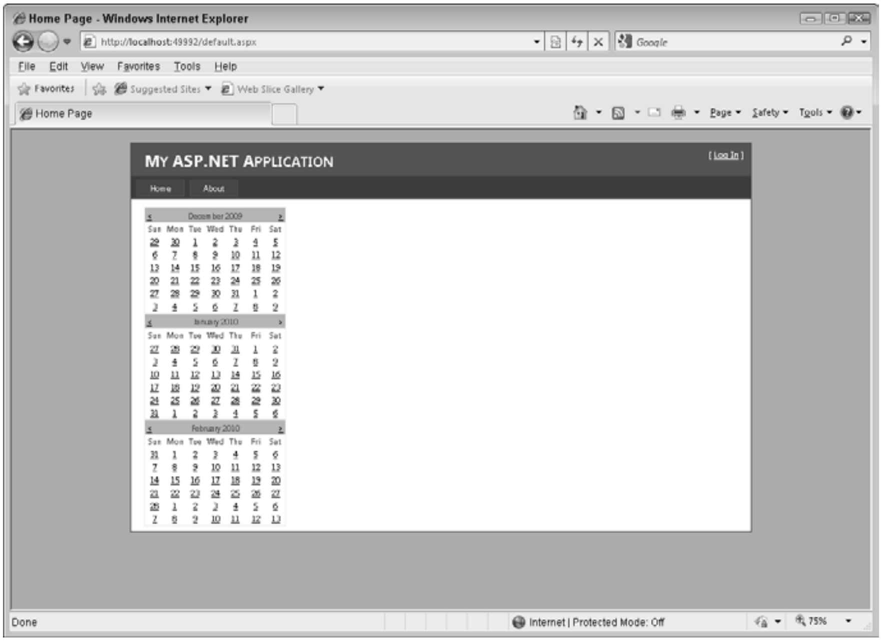
   Visual Studio creates an event handler and opens the Code Editor for you.

10. **Type these two lines of code:**

```
Me.Calendar1.VisibleDate = Me.Calendar2.VisibleDate.AddMonths(-1)
Me.Calendar3.VisibleDate = Me.Calendar2.VisibleDate.AddMonths(1)
```

11. **Press Ctrl+F5 to run your Web site.**

    Your Web page displays the user control with three calendars, as shown in Figure 4-6. As you navigate with the middle control, the other controls are updated, too.


Home Page - Windows Internet Explorer
http://localhost:49992/default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page
Safety
Tools
MY ASP.NET APPLICATION
[Log In]
Home
About
Done
Internet | Protected Mode: Off
75%

< December 2009 >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

< January 2010 >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

< February 2010 >

Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	1	2	3	4	5	6
7	8	9	10	11	12	13

Figure 4-6: Click the middle calendar to update all three calendars.

Changing your view

You can use one of several views to edit ASP.NET Web pages in Visual Studio 2010:

- **Design view:** The `default.aspx` page is opened in Design view by default. Design view is a WYSIWYG visual designer in which you can build your Web page by typing content or dragging and dropping controls on the design surface.
- **Source view:** You can view and edit the underlying HTML markup that's used to build the Web page.
- **Code view:** Display the programming language editor associated with that Web page.
- **Component Designer view:** Create a visual design surface for nonvisual components.

Straight to the Source (view)

Source view in Visual Web Developer displays the HTML markup and ASP.NET syntax used to create the Web page or user control. You use Source view to edit HTML markup and ASP.NET syntax.

You can display Source view of a Web page in one of three ways:

- Click the Source tab at the bottom of the Document Explorer on the Web page.
- Choose View➪Markup.
- Right-click the Web page in Solution Explorer and choose View Markup from the contextual menu.

Source view automatically displays the HTML Source Editing toolbar. It has several commands for formatting the markup and has these two additional validator commands:

- **Target Schema for Validation:** Use this drop-down list to specify whether Visual Studio should use XHTML 1.*x*, HTML 1.0, or a version of Internet Explorer in validating the Web page's markup.
- **Check Page for Accessibility:** The accessibility validator uses a number of accessibility standards to determine whether people with disabilities can use assistive technologies with your page.

Run the validator commands if you want to ensure that your page conforms to industry standards.

When you run your ASP.NET Web page, the content that's sent to the browser isn't what you see in Source view in Visual Studio. Rather, the ASP.NET syntax is converted to HTML.

In this example, the ASP.NET syntax is cleaner than the HTML. Part of the reason is the view state values you see in the HTML. The view state values are shown in the long sequence of characters and represent data stored on the page. View state is added by ASP.NET. In the case of simple HTML elements, such as labels and text boxes, ASP.NET syntax isn't much simpler than HTML. The real difference between HTML and ASP.NET shows up when you use more complex display structures, such as tables. For example, the `GridView` control requires a few lines of ASP.NET syntax and is rendered as a table created by using numerous lines of HTML.

See Book IV, Chapter 7 for information about the view state.

Notice a couple of characteristics about the comparison:

- ASP.NET directives, such as `@Page`, aren't present in the output.
- Any HTML markup with the `runat="`*`server`*`"` attribute is converted to HTML.

For example, this line

```
<form id="form1" runat="server">
```

is converted to this one:

```
<form name="form1" method="post" action="default.aspx" id="form1">
```

The `<form>` tag is special because it designates which controls the users of your Web page can interact with. When you add controls to your Web page by using Design view, the markup necessary to build those controls is placed inside the `<form>` tag.

Source view is a syntax-aware editor that you can use to write or edit HTML and ASP.NET syntax. For example, to add a new hyperlink to your Web page, follow these steps:

1. **Type an opening tag bracket (<).**

   A list of valid HTML and ASP tags appears.

2. **Press the down-arrow key to highlight the `a` tag and then press Tab.**
3. **Press the spacebar to display a drop-down list of valid attributes.**
4. **Select the `href` attribute.**

   To do so, press either the down-arrow key or the letter h. When the `href` attribute is displayed, press Tab to select it.

5. **Type an equal sign (=).**
6. **When the list of available Web pages appears, type** “http://www.cnn.com” **(with the double quotation marks).**

   You can also use the Pick URL option to pick a URL.
7. **Type the closing tag bracket (>).**

   Visual Studio fills in the closing tag for you.

You can also add new attribute/value pairs to existing HTML and ASP.NET tags by pressing the spacebar inside any of the tags.

Code editing

Visual Studio takes care of much of the task of writing code for you. ASP.NET has two coding styles:

- **Inline:** Code is written in Source view by using script tags.
- **Code-behind:** All code is maintained in a separate file.

The default approach uses the code-behind style. The code-behind file is divided into two files:

- One file, generated automatically by Visual Studio, contains all the code generated by Visual Studio to handle the controls you place on the screen and wires up the events you write code for.
- You view the other file when you click in Code view. This file contains all the code you write.

When you run your Web site, the code from these two files is combined.

Earlier versions of ASP.NET had only one code-behind file. This single code-behind file contained all the code generated by Visual Studio and the code written by the developer. Microsoft changed this model into two separate files to keep the code clean and prevent developers from being confused by the generated code. The feature that enables the code to be split into two files is partial classes. Together, the files create a *class,* which is a unit of a program. Partial classes make it possible to split a single class across two files. Partial classes are intended for use with code-generation tools. As a rule, you shouldn’t create your own partial classes. You can read more about using classes to organize your code in Book III, Chapter 3.

You shouldn’t change the code generated by Visual Studio. Any changes you make will likely be overwritten when the file is regenerated. It’s okay to open the generated file from Solution Explorer and view its contents. But you shouldn’t add or edit the code.

Running your Web site

You need to run your Web site from time to time while you're developing to make sure that it's working properly. If you're working mostly with the visual Design view, you shouldn't have many problems. You can run your Web site in one of two modes:

- **Debug mode:** Provides detailed information about errors you might encounter in running your Web site. Most developers execute their Web sites in Debug mode while they're actively developing their sites.
- **Release mode:** After a site is ready to be deployed, its developer usually switches it from Debug mode to Release mode because the site runs faster in Release mode. However, the process halts when an error is encountered, and you don't get much useful feedback.

Most of the sample code in this book uses Release mode because it has less overhead. The downside is that you don't get good feedback when errors occur.

To start your Web site without debugging, press Ctrl+F5. To start it with debugging turned on, press F5. You can also use Debug menu commands for starting your Web site.

Debugging a Web site can be overwhelming — even professionals get hung up. For more information about using and configuring the debugger, see Book III, Chapter 7.

Using Starter Kits

A *starter kit* is a fully functional Web site that you can open and customize to suit your needs. Visual Studio 2010 includes a Personal Web Site starter kit. Starter kits are listed alongside other project templates in the New Web Site dialog box.

You can download additional starter kits from the Downloads page from the ASP.NET site, `www.asp.net`.

Microsoft adds starter kits regularly. Here are some available starter kits:

- **Club Site:** A Web site for a club or organization
- **Time Tracker:** A Web site for tracking hours spent on a project
- **PayPal-enabled eCommerce:** A site for getting started with PayPal
- **DotNetNuke:** A good way to get started with DotNetNuke, which is a Web application framework for creating and deploying Web sites

Chapter 5: Using AJAX for Rich Web Pages

In This Chapter

- Discovering what AJAX does and why you should consider using it
- Using AJAX extensions to create rich Web Applications
- Extending ASP.NET controls with AJAX for smooth page response
- Creating custom AJAX controls for additional functionality

Creating Web pages with rich content while also creating Web pages with nimble response is the Holy Grail of Web development. For years, desktop applications have performed user interface (UI) operations with almost no perceptible delay, but Web applications are at the mercy of the Internet connection speed. Even with a fast Internet connection, though, many Web applications with rich content are sluggish, taking as long as several seconds to load.

This chapter offers a solution: the addition of Asynchronous JavaScript and XML (AJAX) extensions. These extensions go beyond the traditional ASP. NET controls and give you the ability to make your Web pages respond almost as well as desktop applications, even if the content is relatively rich.

AJAX Explained: What It Does and Why You Should Consider Using It

AJAX is a relatively new technology that relies on JavaScript. The JavaScript code that AJAX employs allows it to perform to the point where very little perceptible delay is seen with the UI because AJAX reduces the amount of data that must be transferred from the server to the client. As a result, the user experience is far more enjoyable because users don't wait for page updates and refreshes.

ASP.NET postback architecture

Start with a little background on the ASP.NET postback architecture. Say that you create an ASP.NET Web application that has two text fields: a button and a label. Here's what happens:

1. **The text fields allow users to enter the width and a height of a rectangle.**
2. **The button calculates the area of the rectangle.**
3. **The area is placed into the label showing the result of the calculation.**

If this Web application were a desktop application, the architecture would be trivial. The program code would simply get the values, do the calculation, and then display the result. However, ASP.NET controls execute on the server and not on the client machine. As a result, the page needs to present its data to the server so that the server can do the calculation, populate the label with the result, and serve up the new page. The process of controls executing on the server and returning data to the client is commonly known as a *postback*.

When Microsoft introduced this technology in 2000, developers weren't crazy about it. Postbacks put more load on servers because the calculation and result display require a round trip to the server. Developers soon realized, though, how easy they made Web application development, so the postback architecture won the day.

Unfortunately, though, the round trip to the server causes a delay. Although a desktop application does this trip with no real delay, a Web application may give a noticeable delay during heavy traffic. So, although the ASP.NET postback architecture is brilliant for easy and robust Web development, AJAX, a way to get rid of the page roundtrip delays, was developed.

Partial page updates

You can just about solve the postback delay issue if your applications can simply update the portion of the page that needs to be updated. Easy, right? People have been trying to solve that problem for years with IFrames and other technologies. None, though, have worked nearly as well as AJAX, which is built on JavaScript technology.

AJAX allows Web applications to perform partial page updates. Only the page elements that need to be updated are, thus providing a far more responsive page refresh mechanism. And if a Web application developer is judicious in how pages are designed, almost no perceptible refresh delay is noticed.

AJAX uses JavaScript to perform partial page updates. This section gives a short description of the mechanism.

How JavaScript interacts in HTML documents

First, make sure that you know that JavaScript isn't Java. *Java* is compiled binary code that can reside on the Internet as applets. These applets are loaded and executed via the Java Virtual Machine (JVM).

Comparatively, *JavaScript* can be part of an HTML page. It can technically be a separate file that's referenced in an HTML page, but the browser treats it as if the referenced code is part of the HTML where it's referenced.

The browser interprets JavaScript code. Interpreted code isn't as fast as compiled code, but modern-day browsers and fast machines easily handle the interpretation so that users don't notice.

JavaScript has been used for many years in a technique known as *Dynamic HTML* (DHTML), which is a way of making HTML more dynamic by allowing the client machine to act on HTML elements. For example, DHTML code can cause a test string to turn green when the mouse hovers over it and then back to black when the mouse leaves its area.

Here's a simple JavaScript page that changes the color of a text string when the mouse hovers over it:

```
<html>
<body>
<h1>This page demonstrates DHTML.</h1>
<span id="Test"
  onMouseOver="this.style.color='green'"
  onMouseOut="this.style.color='black'">
    Hi there, hover over me.</span>
</body>
</html>
```

If you're unsure how to work with JavaScript, you can check out this tutorial:

```
www.w3schools.com/js/default.asp
```

Using the JavaScript HttpRequest mechanism

DHTML isn't enough to do partial page updates. You still need a way to make a request from the server. The mechanism that's used for this is an HTTP request, performed by the JavaScript `XMLHttpRequest` object.

With an HTTP request, a Web page can make a request to — and get a response from — a Web server without reloading the page. The user will stay on the same page, not noticing that scripts might request pages or send data to a server in the background.

Listing 5-1 performs a simple fetch of a document via an HTTP request that relies on the `XMLHttpRequest` object.

Listing 5-1: Simple Document Fetch Using XMLHttpRequest

```
<script language="JavaScript">

var XMLHttp;

function LoadData(url)
{

     XMLHttp = null;

     // code for Firefoxetc.
     if( window.XMLHttpRequest )
     {
          XMLHttp= new XMLHttpRequest();
     }

     // code for Internet Explorer
     else if( window.ActiveXObject )
     {
          XMLHttp= new ActiveXObject( "Microsoft.XMLHTTP" );
     }

     if( xmlhttp != null )
     {
          XMLHttp.onreadystatechange = state_Change;
          XMLHttp.open( "GET", url, true );
          XMLHttp.send( null );
     }
     else
     {
          alert( "Your browser does not support XMLHTTP." );
     }

}
</script>
```

Fortunately, you never need to do this kind of JavaScript coding. The AJAX extensions do it all for you. The AJAX controls inject the appropriate JavaScript code into the HTML output stream without you needing to code any JavaScript yourself.

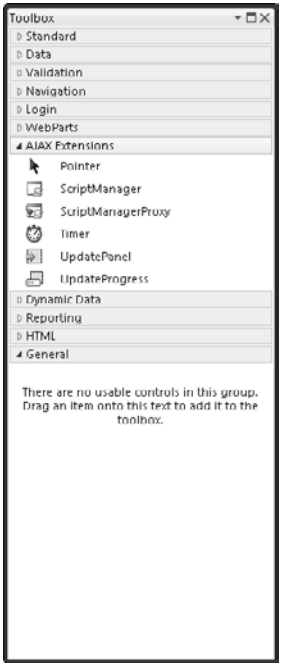
Using the XML Data that is returned

The data that's returned is in XML format. XML is simply a way of representing data. It's done with tags, the same way that HTML is done, except that XML allows you to create your own tags that best describe and represent your data.

Here again, you may be worried about handling the XML. Converting simple XML to non-XML data isn't hard, but it's an extra step that can complicate things. Fortunately, the AJAX controls handle all this conversion for you. Instead of actually getting XML, your controls give you simple data types, such as strings and integers.

Using AJAX Extensions

AJAX extensions are included in Visual Studio 2010. In Visual Studio 2005, you had to download and install them, but these extensions are part of the Visual Studio 2010 installation. You can find the AJAX extensions in the toolbox, in the section labeled AJAX Extensions, as shown in Figure 5-1.



There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

Figure 5-1: The AJAX extensions are in the toolbox.

Creating your first AJAX Web application

Get started with AJAX by creating your first AJAX Web site. To create a new AJAX-enabled Web site, follow these steps:

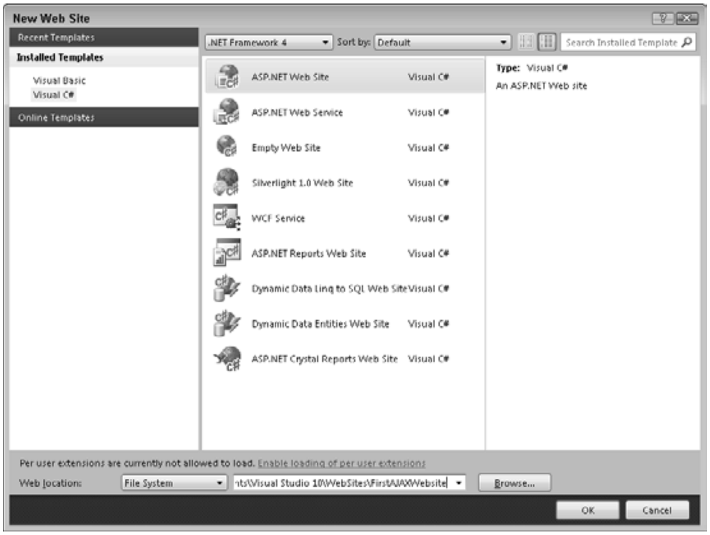
1. **Choose File⇨New Web Site.**

   The New Web Site dialog box appears.

2. **Click the ASP.NET Web Site template.**
3. **Select the language from the Language drop-down list.**
4. **Choose HTTP for the location for the new Web site. Enter a name, such as FirstAJAXWebsite, after the pathname, as shown in Figure 5-2.**

Book IV
Chapter 5

Using AJAX for Rich Web Pages



.NET Framework 4 Sort by: Default

ASP.NET Web Site	Visual C#
ASP.NET Web Service	Visual C#
Empty Web Site	Visual C#
Silverlight 1.0 Web Site	Visual C#
WCF Service	Visual C#
ASP.NET Reports Web Site	Visual C#
Dynamic Data Linq to SQL Web Site	Visual C#
Dynamic Data Entities Web Site	Visual C#
ASP.NET Crystal Reports Web Site	Visual C#

Type: Visual C#

An ASP.NET Web site

Per user extensions are currently not allowed to load. Enable loading of per user extensions

Web location: File System nts\Visual Studio 10\WebSites\FirstAJAXWebsite

Figure 5-2: Name the Web site.

5. **Click OK to create the ASP.NET Web site.**

   The `default.aspx` page opens in Split view.

6. **From the AJAX Extensions tab of the toolbox, place one — and only one — `ScriptManager` object into the form in the Design view.**

That's all you need to do to create an AJAX-enabled Web site. Continue to explore how Ajax sites work by adapting and modifying this basic application in the sections that follow.

Adding traditional ASP.NET controls to the form

Controls in a Web site are similar to controls in applications that you create in Visual Studio — the buttons, boxes, and so on that users interact with onscreen. The easiest way to figure out how they work is to experiment with them. Using the site you created in the preceding section, you can start by adding three labels and a button to the site. These are traditional ASP.NET controls. Get these controls working and take a look at this simple application first, though, before going on to implement a partial page update pattern.

To add the ASP.NET server controls, follow these steps:

1. **Add three labels and a button to the form.**
2. **Double-click the button to create an event that causes a postback.**
3. **Double-click the page and create a `Page_Load` event handler.**

4. **Add this code to the Page_Load method:**

   - **VB**

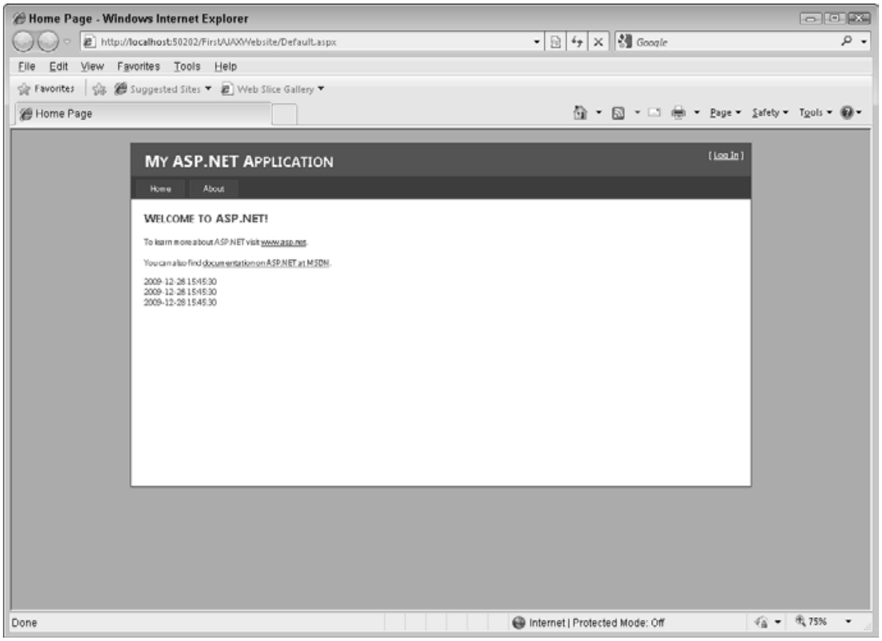
     ```
     Label1.Text = Date.Now
     Label2.Text = Date.Now
     Label3.Text = Date.Now
     ```

   - **C#**

     ```
     Label1.Text = DateTime.Now.ToString();
     Label2.Text = DateTime.Now.ToString();
     Label3.Text = DateTime.Now.ToString();
     ```

5. **Run the program by pressing Ctrl+F5.**

   You see the labels reflect the time of the last postback, as shown in Figure 5-3. The button causes postbacks and subsequent time changes.


Home Page - Windows Internet Explorer
http://localhost:50202/FirstAJAXWebsite/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page Safety Tools
[Log In]
Home
About
Done
Internet | Protected Mode: Off
75%

MY ASP.NET APPLICATION

WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

You can also find documentation on ASP.NET at MSDN.

2009-12-28 15:45:30
2009-12-28 15:45:30
2009-12-28 15:45:30

Figure 5-3: The labels on the form show the current date and time.

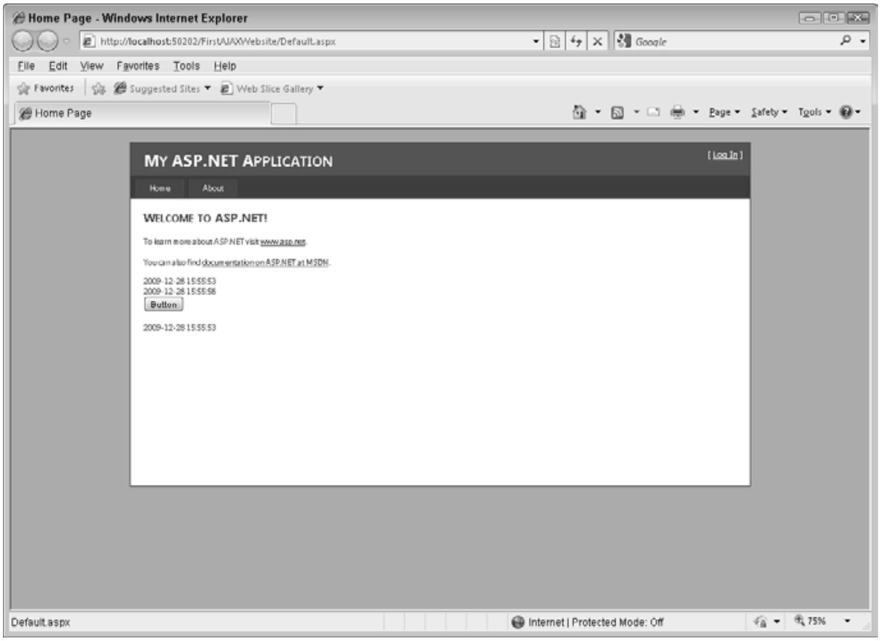
Using AJAX to implement a partial page update pattern

After you have an ASP.NET application working, use AJAX to implement a partial page-update pattern. Working from the site created in the preceding section, the end result is that the button causes an update for the second label only.

To implement the partial page update pattern, follow these steps:

1. **From the AJAX Extensions tab of the toolbox, add an `UpdatePanel` control above the second label.**
2. **Drag the second label into the `UpdatePanel` control.**
3. **Drag the button into the `UpdatePanel` control.**
4. **Run the program by pressing Ctrl+F5.**

   You see the labels reflect the time of the first page load. Then, only the second label is updated in response to a button click, as shown in Figure 5-4.



Home Page - Windows Internet Explorer
http://localhost:50202/FirstAJAXWebsite/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page Safety Tools
Default.aspx
Internet | Protected Mode: Off
75%

MY ASP.NET APPLICATION

[Log In]

Home About

WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

You can also find documentation on ASP.NET at MSDN.

2009-12-28 15:55:53
2009-12-28 15:55:58

2009-12-28 15:55:53

Figure 5-4: The button causes an update in the second label only.

Updating controls outside UpdatePanel

Sometimes, you need to do partial page updates when the control that fires off the update can't be inside `UpdatePanel`. Modify your application from preceding sections once again and take the button out of `UpdatePanel`. Then, add a `Triggers` collection to `UpdatePanel`. A `Triggers` collection manages the AJAX events that fire off in response to UI events. In the `Triggers` collection, specify that the second label should be updated by the button, even though the button is outside `UpdatePanel`.

To implement the Triggers collection and have a button outside UpdatePanel cause an update, follow these steps:

1. **From the FirstAJAXWebSite that you're working with, move the button from inside UpdatePanel to the bottom of the page so that it's no longer inside UpdatePanel.**
2. **Open the Properties window for UpdatePanel.**
3. **Click the ellipsis button beside the Triggers property.**

   The Collection Editor dialog box appears, as shown in Figure 5-5.

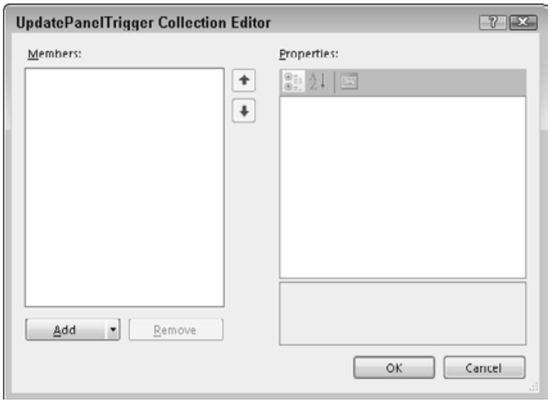


Figure 5-5: The Collection Editor allows you to add a trigger.

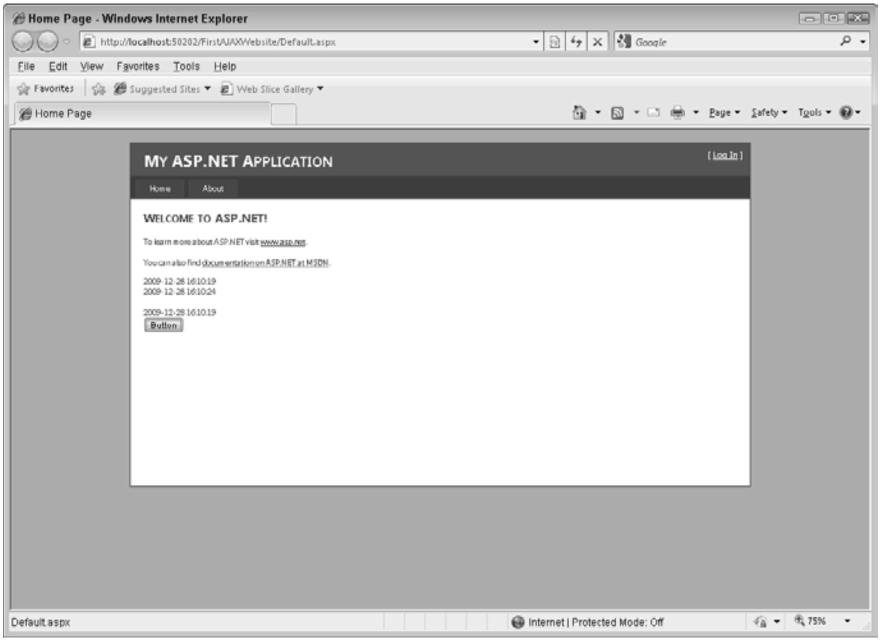
4. **Click the Add button.**

   An AsynPostBack member is added.
5. **In the right side of the editor, set the controlID to Button1 and the EventName to Click.**

   This step wires up the buttons Click event to the UpdatePanel update so that the Click event fires off an event trigger that connects it to UpdatePanel.
6. **Click Add.**

   An AsynPostBack member is added.
7. **Run the program by pressing Ctrl+F5.**

   The labels reflect the time of the first page load. Then, only the second label is updated in response to a button click, even though it's no longer in UpdatePanel, as shown in Figure 5-6.


Home Page - Windows Internet Explorer
http://localhost:50202/FirstAJAXWebsite/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page
Safety
Tools
[Log In]
Home
About
Button
Default.aspx
Internet | Protected Mode: Off
75%

MY ASP.NET APPLICATION

WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

You can also find documentation on ASP.NET at MSDN.

2009-12-28 16:10:19
2009-12-28 16:10:24

2009-12-28 16:10:19

Figure 5-6: The button causes an update only in the second label.

Using the AJAX Controls

There's more to AJAX than the extensions. A rich set of controls is provided in the AJAX Control Toolkit, which you can download from `www.asp.net`. At this Web site, click the AJAX button, click the Download button, and then follow the links to download the latest AJAX Control Toolkit.

The Toolkit is contained inside a Zip archive, so extract it to your hard drive. For this explanation, use this extraction path:

```
C:\Users\lux mundi\Documents\Visual Studio 2010 Desk Reference\
    AjaxControlToolkitBinary
```

Adding the AJAX Control Toolkit to the Visual Studio toolbox

Before using the AJAX Control Toolkit, you need to add a reference to the Visual Studio toolbox. This reference makes dragging controls to the Web application's form easy.

To add the AJAX Control Toolkit to the Visual Studio toolbox, follow these steps:

1. **Open the Visual Studio toolbox.**
2. **Right-click and choose Add Tab.**
3. **Open the Properties window for `UpdatePanel`.**
4. **Name the tab** AJAX Toolkit**.**
5. **Right-click the newly created tab and choose Choose Items.**
6. **Click the Browse button and navigate to the file at**

   ```
   C:\Users\lux mundi\Documents\Visual Studio 2010 Desk Reference\
       AjaxControlToolkitBinary\AjaxControlToolkit.dll
   ```

   You see many controls appear in the AJAX Toolkit tab, as shown in Figure 5-7.

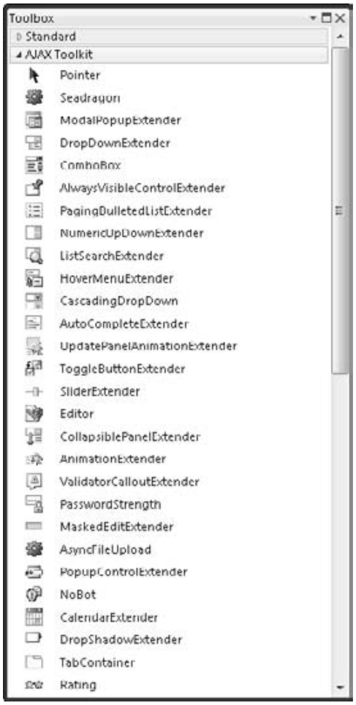


Figure 5-7: The AJAX Toolkit has many controls.

Using AJAX controls from the Toolkit

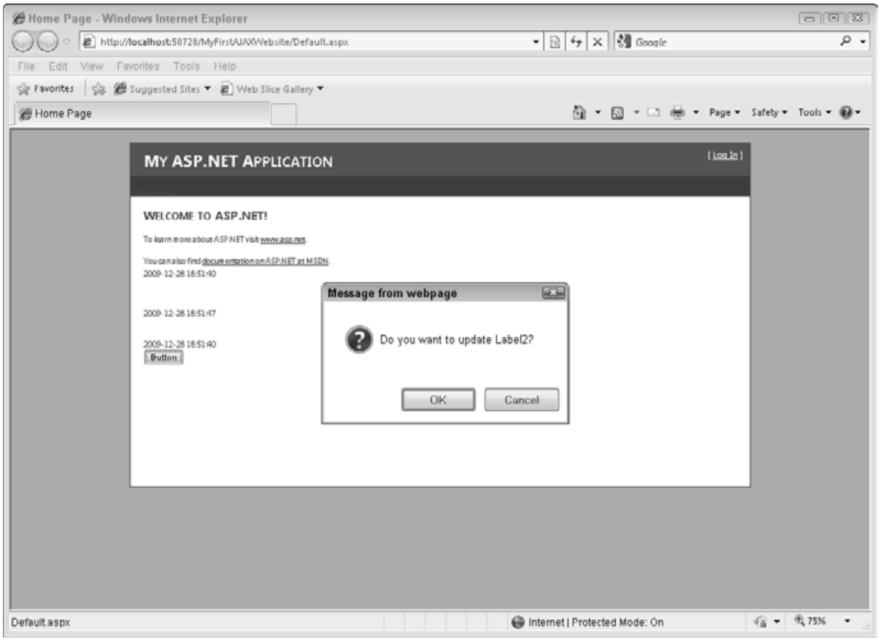
After you have the AJAX Toolkit installed and set up, as explained in the preceding section, you can get started using the controls. Begin by using a `ConfirmButtonExtender`. The control gives you the ability to confirm actions in response to user button clicks.

To use a `ConfirmButtonExtender`, follow these steps:

1. **Drag a `ConfirmButtonExtender` control from the AJAX Toolkit tab area of the toolbox onto the form right above the button.**
2. **Open the `ConfirmButtonExtender` properties.**
3. **Choose `Button1` as the `TargetControlID`.**
4. **Enter some text into the `ConfirmText` property.**

   For example, enter Do you want to update the second label?
5. **Run the program by pressing Ctrl+F5.**

   When you click the button, you get a confirmation alert, as shown in Figure 5-8.


Home Page - Windows Internet Explorer
http://localhost:50728/MyFirstAJAXWebsite/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page
Safety
Tools
[Log In]
Button
OK
Cancel
Default.aspx
Internet | Protected Mode: On
75%

MY ASP.NET APPLICATION

WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

You can also find documentation on ASP.NET at MSDN.

2009-12-28 18:51:40

2009-12-28 18:51:47

2009-12-28 18:51:40

Message from webpage

Do you want to update Label2?

Figure 5-8: This AJAX control confirms a button action.

Chapter 6: Laying Out Your Web Site

In This Chapter

- **Validating user input**
- **Adding navigation controls to your site**
- **Using themes and master pages**
- **Laying out Web pages with Cascading Style Sheets (CSS)**

Look at any Web site, and you're likely to see common elements, such as form-validation feedback, navigational aids, and consistent headers and footers. In previous versions of Visual Studio, you had to be conversant in technologies as varied as JavaScript and XML to provide these features. Thanks to many of the new features in Visual Studio and ASP.NET, you can create high-quality Web sites while writing a minimal amount of code.

This chapter gets you acquainted with themes, skins, and master pages and introduces you to the benefits of using CSS to lay out your Web pages. You can also see how to use the ASP.NET validation controls to test user input and provide feedback to users.

Keeping Track of Input with Validation Controls

After you create only a few Web forms, you'll realize that you need some way to make sure that all your check boxes are checked and your text boxes are filled with the right information. ASP.NET has a collection of validation controls to help you do just that.

Validation controls provide validation services for the controls you place on a Web page, ensuring that the data that's entered is in the correct format. The validation controls also enable you to give users feedback about the correct data. For example, a Web page can include text boxes to capture these input types from users:

- E-mail address
- Phone number
- Date of birth

How can you be sure that the values that are entered are in the proper formats? Further, how can you ensure that values are even entered?

You have three options:

- **Do nothing and hope for the best.**
- **Write server-side code that validates your text boxes after visitors click the Submit button.** This process requires a round trip to the server. You can read more about these round trips, usually called *postbacks,* in Book IV, Chapter 4.
- **Write client-side script that validates your text boxes as users enter their data.**

Whenever possible, using client-side script for form validation is the preferred approach because it eliminates a trip to the server and provides immediate feedback to visitors. You can write all this client-side script yourself, or you can let the ASP.NET validation controls write it for you.

You can use multiple validation controls for form fields. For example, you may need to check a form field to make sure that it has data in it and at the same time check to make sure that it's identical to another form field. In such a case, you can use two validation controls for one form field. ASP.NET provides these validation controls:

- `RequiredFieldValidator`: Ensures that required fields contain a value.
- `RangeValidator`: Checks whether entries fall within a specified range, such as between start and end dates.
- `RegularExpressionValidator`: Tests whether entries match a certain pattern, such as ZIP + 4 or a telephone number.
- `CompareValidator`: Compares an entry with a value.
- `CustomValidator`: Lets you write your own control when none of the other controls provides the level of validation you need.
- `ValidationSummary`: Displays a summary of all validation errors.

Adding a validation control

Using validation controls is straightforward. For example, to require that a user fill in a text box on an existing Web page, follow these steps:

1. **Open (or create) the Web page to which you want to add validation controls.**

   See Book IV, Chapter 4 for more information about creating Web pages.

2. **Drag and drop a `Label` control and a `TextBox` control onto the Web page.**

3. **Set the `Text` property on the label to `Enter Name` and set the text box `ID` property to `txtEnterName`.**
4. **Drag and drop a `RequiredFieldValidator` control from the Validation section of the toolbox.**
5. **Set the following properties for the `RequiredFieldValidator` control from the Properties window:**
   - `ID`: Type **EnterNameRequired.**
   - `ControlToValidate`**:** Select txtEnterName from the drop-down list. It's the text box control you create in Step 3.
   - `Text`: Type **Name is a required field**.
6. **Add a `Button` control and a `Label` control to the page.**
7. **Set the label's `ID` property to `lblConfirmation`, its `Text` property to `Confirmation Message`, and the button control's `Text` property to `Submit`.**
8. **Double-click the button.**

   The button's `Click` event appears in the Code Editor.
9. **Type this code in the button's `Click` event:**
   - **VB**

     ```
     If Page.IsValid Then
         lblConfirmation.Text = "Page is valid! "
     End If
     ```

   - **C#**

     ```
     if( Page.IsValid )
     {
         lblConfirmation.Text = "Page is valid!";
     }
     ```

   This code snippet displays the message `Page is valid!` on the Confirmation label when the user clicks the button.

TIP

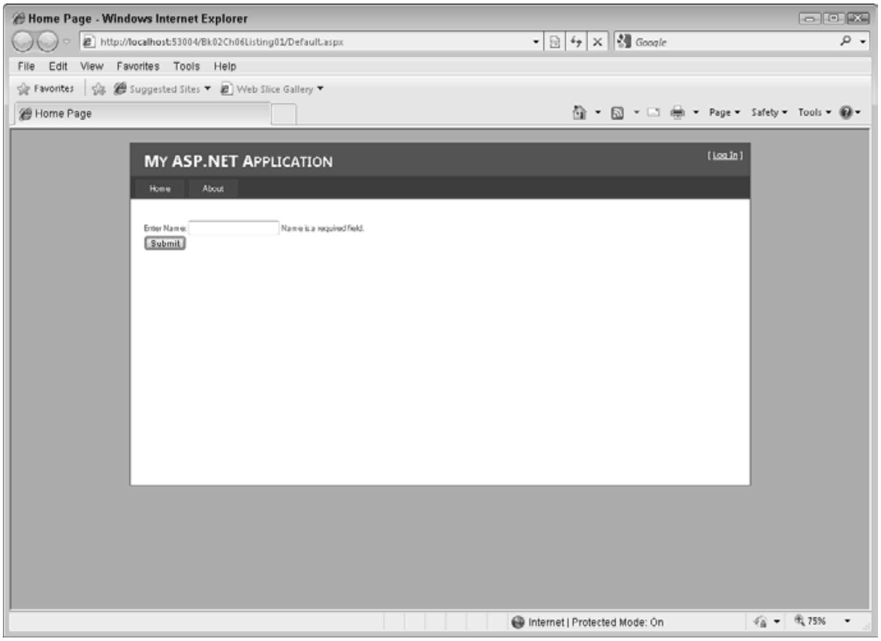
See Book III, Chapter 4 to read more about using events.

Testing a validation control

To test your validation control, follow these steps:

1. **Press Ctrl+F5 to run your site.**
2. **Click the Submit button on the page.**

   The page displays `Name is a required field` next to the text box, as shown in Figure 6-1.


Home Page - Windows Internet Explorer
http://localhost:53004/Bk02Ch06Listing01/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page
Safety
Tools
MY ASP.NET APPLICATION
[Log In]
Home
About
Internet | Protected Mode: On
75%

Enter Name: Name is a required field.

Submit

Figure 6-1: This error message is displayed next to the required text box.

The `RequiredFieldValidator` control uses client-side scripting to test the contents of the text box. When you click the Submit button, the browser doesn't call the server. Instead, a client-side script validates the control. If the control passes the validation, data is sent to the server. Using client-side scripting is more efficient because you don't make wasted trips back to the server just to see whether the user has filled in a text box.

You should make a trip to the server whenever you need to validate sensitive information, such as a password. Using a validation control with client-side scripting isn't appropriate because you have to store the password in the client-side script. Client-side scripts are visible to users, so you would compromise your password. See Book IV, Chapter 6 to see how to use the new ASP.NET `Login` controls to validate passwords.

3. **Type a name in the text box and then press Tab.**

   The error message goes away.

4. **Delete the name from the text box and then press Tab.**

   The message appears again.

5. **Type a name in the text box again and click the Submit button on the page.**

   The button's `Click` event is sent to the server for processing because the text box has a value in it, as required by the `RequiredFieldValidator` control.

6. **The page displays `Page is valid!` next to the button.**

   You may wonder why the `Page is valid!` message wasn't displayed earlier, when you typed a name in the text box in Step 3 and the error message disappeared. For the message to appear, the data must be sent to the server. The data can be sent only if the validation control confirms that the data is valid when the button is clicked.

You can download a project for the preceding example from the companion Web site for this book at `www.dummies.com/go/vs2010`.

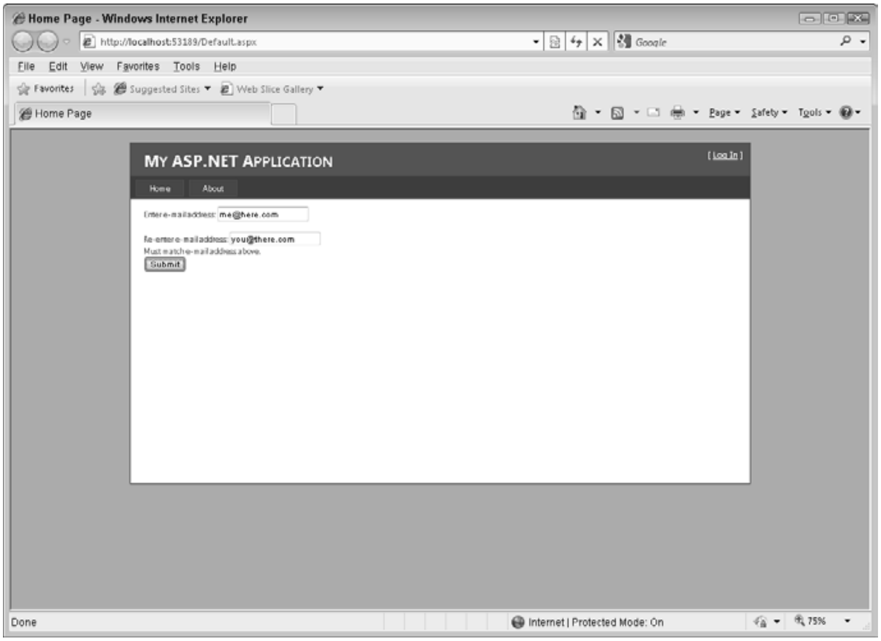
Handling multiple validation controls

You can also mix and match validators. Often, multiple validations must be performed on a single control. For example, you may want to test whether a control has a value and whether the value conforms to a certain pattern, such as a phone number. In this case, you use a `RequiredFieldValidator` and a `RegularExpressionValidator`. The following example walks you through using the comparison and pattern validators to validate the same control:

1. **Drag and drop a label control and text box control to your Web page; set the label's `Text` property to `Enter e-mail address` and set the text box `ID` to `txtEmail1`.**
2. **Drag and drop another label control and text box control onto your Web page; set the label's `Text` property to `Re-enter e-mail address` and set the text box `ID` to `txtEmail2`.**
3. **Add `RequiredFieldValidator` controls for each of the text boxes; set the properties for the controls as described in the earlier section "Adding a Validation Control."**
4. **Add a `RegularExpressionValidator` control to the page.**
5. **Set the control's properties as follows:**
   - `ValidationExpression`: Click the ellipsis and choose Internet E-mail Address.
   - `Text`: Enter **The e-mail address must be properly formatted (for example, foo@bar.com)**.
   - `ControlToValidate`: Select txtEmail1 from the drop-down list.
6. **Add a `CompareValidator` control to the page.**
7. **Set the control's properties:**
   - `ControlToCompare`: Select txtEmail1.
   - `ControlToValidate`: Select txtEmail2.
   - `Text`: Type **Must match e-mail address above**.

8. **Add a button control, setting the `Text` property to `Submit`.**
9. **Press Ctrl+F5 to run your site and test the validators.**

   Your page should look similar to the one shown in Figure 6-2.


Home Page - Windows Internet Explorer
http://localhost:53189/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page Safety Tools
My ASP.NET Application
[Log In]
Home
About
Submit
Done
Internet | Protected Mode: On
75%

Enter e-mail address: me@here.com

Re-enter e-mail address: you@there.com

Must match e-mail address above.

Figure 6-2: The validators display error messages next to the text boxes.

You can download a project for the preceding example from the companion Web site for this book at `www.dummies.com/go/vs2010`.

Notice how the error message for the `RegularExpressionValidator` doesn't align next to the text box. That's because a placeholder for `RequiredFieldValidator` is sitting between the text box and the `RegularExpressionValidator` error message, as in this abbreviated code:

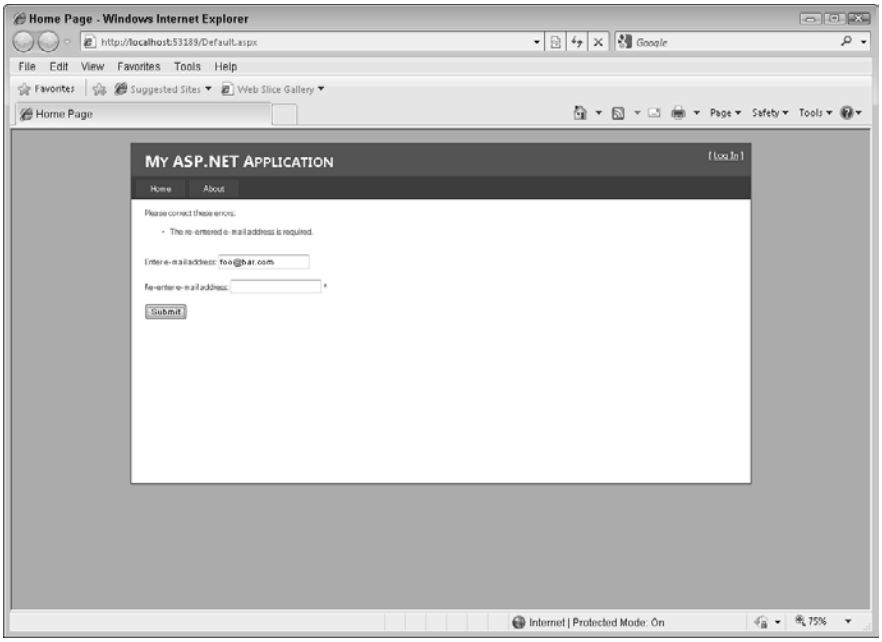
```
<asp:TextBox ID="txtEmail1" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
 ErrorMessage="E-mail is a required field." ForeColor="Red"
 ControlToValidate="txtEmail1" Display="Dynamic"></asp:RequiredFieldValidator>
        <br />
        <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
        runat="server"
        ErrorMessage="The e-mail address must be properly formatted (e.g. foo@
    bar.com)"
        ControlToValidate="txtEmail1" ForeColor="Red"
        ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"></
    asp:RegularExpressionValidator>
        <br />
```

To eliminate the placeholder, change the Display property on RequiredFieldValidator from Static to Dynamic. Now, your error message flows with the page layout.

Instead of displaying your messages next to text boxes, you can display them in a message box on the page. Follow these steps:

1. **Add a ValidationSummary control to the top of your page.**
2. **Type** Please correct these errors: **in the HeaderText property.**
3. **Set the ErrorMessage property on each of your validation controls to the message you want to display in the ValidationSummary control.**
4. **Put an asterisk in the Text property of each of your validation controls.**
5. **Press Ctrl+F5 to run your site and test it.**

The ValidationSummary control displays all error messages at the top of the page, as shown in Figure 6-3.


Home Page - Windows Internet Explorer
http://localhost:53189/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page
Safety
Tools
My ASP.NET Application
[Log In]
Home
About
Submit
Internet | Protected Mode: On
75%

Please correct these errors:

- The re-entered e-mail address is required.

Enter e-mail address: foo@bar.com

Re-enter e-mail address: *

Figure 6-3: Your site should look similar to this one.

Now, the user receives all error messages at the top of the screen after clicking the Submit button. An asterisk is displayed next to the text boxes, which makes an interesting demonstration of the ErrorMessage and Text properties:

- Use the `ErrorMessage` property, along with the `ValidationSummary` control, when you want to display all feedback in a single place on the screen at one time.
- Use the `Text` property to display immediate feedback next to the control being validated.

Combining these approaches allows the user to see a visual cue next to the text box in question. These properties can accept text, HTML, or images.

You can group validation controls by using the `ValidationGroup` property. You can then use, for example, two `ValidationSummary` controls on a single page.

Mapping Out Your Site Navigation

ASP.NET provides built-in support for Web site navigation by using a combination of a site map and a set of navigation controls driven by the site map.

A *site map* is a logical description of the layout of a Web site. Visual Studio includes the following navigation controls:

- `Menu`
- `SiteMapPath`
- `TreeView`

Adding a site map

ASP.NET supports an XML-file-based site map by default. You can use a database-driven site map, if you choose. The syntax for the site map uses these two nodes:

- `<sitemap>`: The root node of the sitemap document contains all the `<sitemapnode>` nodes.
- `<sitemapnode>`: Each node represents a page on your Web site. Nodes can be nested.

The `<sitemapnode>` node has these attributes:

- `Description`: The page description
- `Title`: The page title
- `URL`: The URL for accessing the page

You can also specify a collection of roles that can access `<sitemapnode>` and a collection of custom attributes. These roles are likely to be used with a custom site map provider.

To create a new site map for an existing Web site, follow these steps:

1. **Right-click the Web site's project folder in Solution Explorer and choose Add New Item from the contextual menu.**

   The Add New Item dialog box appears.

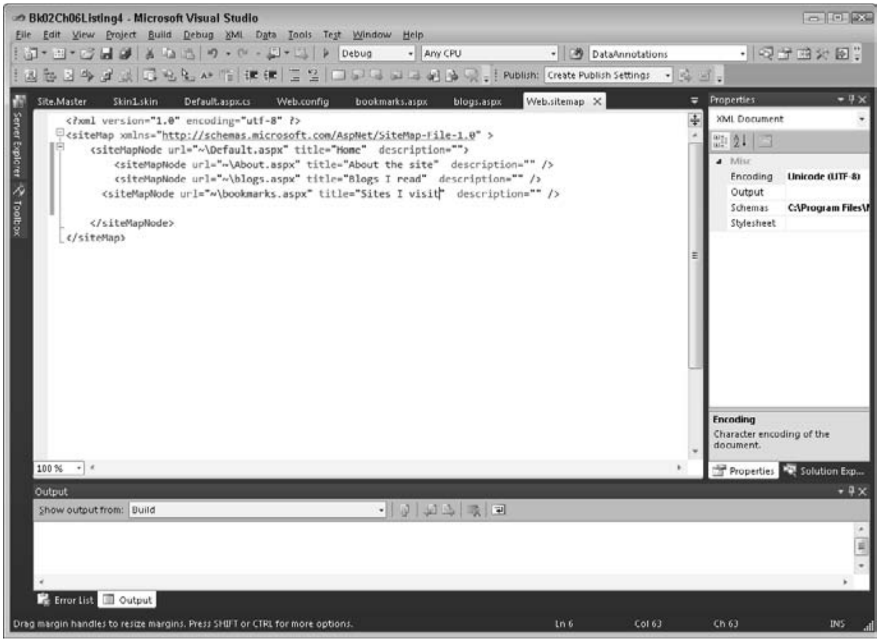
2. **Click the Site Map icon.**

   Don't change the name of the site map.

3. **Click the Add button to add the site map to your Web site.**

   Visual Studio opens a new site map file in Document Explorer.

4. **Type the URL, title, and description in the `siteMapNode` tags for each page in your Web site, as shown in Figure 6-4.**



```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="~\Default.aspx" title="Home"  description="">
        <siteMapNode url="~\About.aspx" title="About the site"  description="" />
        <siteMapNode url="~\blogs.aspx" title="Blogs I read"  description="" />
      <siteMapNode url="~\bookmarks.aspx" title="Sites I visit"  description="" />

    </siteMapNode>
</siteMap>
```

Figure 6-4: Type your information in the site map file.

You can add as many `siteMapNode` tags as required. `siteMapNodes` can be nested to create a tree of nodes.

Suppose that you have a Web site with the following four pages:

- `default.aspx`
- `about.aspx`

- blogs.aspx
- bookmarks.aspx

The following bit of code creates a site map you can use to describe access to the four pages:

```
<siteMapNode url="~/default.aspx" title="Home">
   <siteMapNode url="~/about.aspx" title="About the site" />
   <siteMapNode url="" title="My Research Page">
     <siteMapNode url="~/blogs.aspx" title="Blogs I Read" />
     <siteMapNode url="~/bookmarks.aspx" title="Sites I Visit" />
   </siteMapNode>
</siteMapNode>
```

Adding navigation controls

To do something meaningful with the site map, you must include it with one of the navigation controls to display its contents. To add a `TreeView` control to the `default.aspx` page, for example, follow these steps:

1. **Drag and drop the `TreeView` control from the Navigation tab of the toolbox into Design view of the `default.aspx` page.**

   An image of the `TreeView` control appears, along with a list of tasks.

   If the list of tasks isn't displayed, click the glyph in the upper-right corner of the `TreeView` control.

2. **In the TreeView Tasks list, click the drop-down arrow for Choose Data Source and select <New Data Source> from the drop-down list.**
3. **In the Data Source Configuration Wizard, click the Site Map icon.**
4. **Click OK.**

   The tree view is updated to reflect your site map, as shown in Figure 6-5.

5. **Run your Web site by pressing Ctrl+F5 and test the hyperlinks in your `TreeView` control.**

Managing Your Site's Appearance with Themes and Master Pages

Visual Studio and ASP.NET provide two tools for managing the look and feel of your Web sites:

- **Themes and skins:** Easily separate the appearance attributes of controls from the controls themselves.
- **Master pages:** Create a single page template to use with all your content pages.

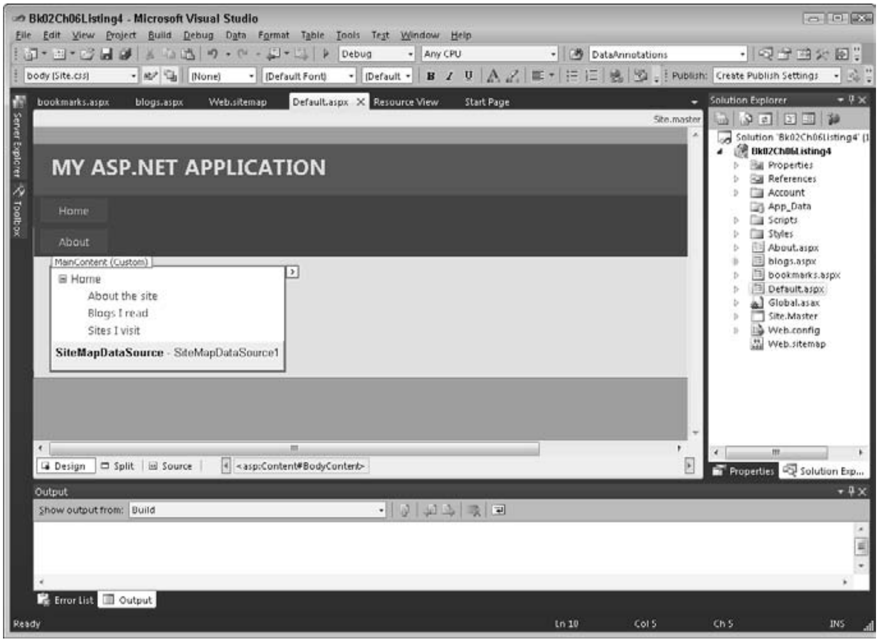


Figure 6-5: The `Tree View` control gets its data from the site map.

Themes and master pages make administering your Web site easier. When you use themes and master pages, visitors can personalize their user experience on your Web site. For more information on personalization, see Book IV, Chapter 7.

Using themes and skins

A *theme* defines the look and feel of your Web site. Themes include these elements:

- **Skins:** A set of properties that define a control's appearance
- **CSS:** A standard for defining the appearance and layout attributes of HTML elements

  TIP

  See the upcoming section "Laying Out Web Pages with CSS" for more information about creating Cascading Style Sheets.
- **Images:** Files that define the site's appearance, such as a company logo

A theme has, at minimum, a skin file. Some overlap occurs between skins and CSS because both are used to control appearance. They have some crucial differences, however:

Cascading styles

A *CSS style* is a set of attributes that defines the appearance and layout of an HTML element. For example, you can create a CSS style that sets a paragraph to use a specific font and font size. Think of a Web page as a container that holds containers of HTML elements. Paragraphs and `div` tags are examples of containers that hold content and other HTML elements.

You can define styles for each container (such as a Web page or a paragraph). As you define styles, the styles closest to the content are applied. For example, you may define a style that sets the font color to black and the font family to Arial at the Web-page level. If you define another style at the paragraph level that sets the font color to red, the content inside the paragraph is red. The font family is Arial, as defined for the Web page, unless you define a different font family for the paragraph.

In this way, styles *cascade* — like a waterfall — from higher-level containers to lower-level containers. The style elements defined closest to the content win over style elements defined in higher containers.

- **Skins don't cascade.** Unlike with CSS, where you can create a hierarchy of styles, you define one skin for each type of control you use on your site. See the sidebar "Cascading styles" for more details on how CSS styles cascade.
- **Skins define properties for ASP.NET Web server controls.** CSS styles apply to HTML elements.
- **Skins apply to a single Web server control.** Styles can apply to a single HTML element or to collections of elements.
- **You can use styles to control the positioning of elements on a page.**

Here are some guidelines for choosing between skins and styles.

- **Use skins when**
  - You're setting properties specific to Web server controls.
  - You want to provide personalization features for your visitors. For example, you might want to display images specific to users or provide color themes.
- **Use styles to**
  - Set site-wide or page-wide properties, such as fonts.
  - Control page layout.

TIP

Use skins to define general properties. If you need to define the appearance of a single element on a page, use styles.

Adding a theme to an existing site

You define themes for individual Web pages, an entire Web site, or all sites hosted on a Web server.

To add a theme to a Web site and associate the theme with the site, follow these steps:

1. **Right-click the Web site's project folder in Solution Explorer and choose Add ASP.NET Folder⇨Theme.**

   A theme folder is added to the project.

2. **Enter a name, such as MyTheme, for the theme folder.**

   You can rename themes by right-clicking and choosing Rename from the theme folder's contextual menu.

3. **Double-click the `web.config` file in Solution Explorer to open it.**

   All your Web site configuration information is stored in the `web.config` file.

4. **Add the theme attribute to the page node in the `web.config` file so that the node looks like this:**

   ```
   <pages theme="MyTheme">
   ```

5. **Close the `web.config` file and save the changes when you're prompted.**

   All pages on your Web site now use the skins and the style sheet you add to the MyTheme folder.

Adding a theme as a style sheet

You can also apply a theme as a style sheet. A *style sheet theme* cascades like a style sheet, which means that local page settings take precedence over the theme. To set the Web site to use a style sheet theme, use this code instead of the code in Step 4 in the preceding set of steps:

```
<pages StyleSheetTheme="MyTheme">
```

To apply a theme to an individual page on your Web site, specify the proper attribute on the page's directive line, like this:

```
<%@ Page Theme="MyTheme" %>
<%@ Page StyleSheetTheme="MyTheme" %>
```

To apply a theme to all sites on a Web server, add a Themes folder to the path:

```
iisdefaultroot\aspnet_client\system_web\version\
```

Refactoring: Trying a new approach

Going through and changing your approach to solving a problem is *refactoring.* Although this process is often used in the context of building software libraries, it applies to any situation in which you have to make design choices. Choosing to use skins may make perfect sense today, but you may decide to implement style sheets later when you're upgrading the site. The point of refactoring is to allow yourself the freedom to make a decision now with the understanding that you can change your mind in the future.

Add your themes to the Themes folder. You can also add the Theme attribute to the pages element in the Web server's `machine.config` file.

Defining skins

Before you can start defining skins, think about the steps for creating your page layout and design:

1. **Identify all common elements that you expect to use repeatedly throughout your site.**

   You want to think in abstract terms about how many headings you want and how each one is different from the others. Picture your content in sections, such as sections for searches, What's New information, RSS (really simple syndication) feeds, and banner ads.

2. **Decide which pages, if any, are unique.**

   For example, a site commonly has a unique home page, and then individual pages reuse common elements.

3. **Decide whether you want to use skins or styles to control the site's appearance.**

   You can always change your mind.

4. **Implement your skins and styles.**

5. **Test your layout frequently.**

   Don't be afraid to change your mind.

Try not to get hung up on whether you should implement a skin or a style. Folks who have used styles for a long time sometimes lean toward using them. Remember that your goal is to have working software. Don't fret over best-practices implementation. Go with what works and don't be afraid to change your approach later.

The following example assumes that you're creating a Web site with these elements:

- A header to place elements such as your site's name and logo
- A sidebar on the left side for navigation
- A footer to place elements such as copyright information
- A main content area

The main content area is the part of the screen that changes dynamically. To define skins for both recurring content and one-off elements, suppose you identify that you need these labels:

- A Web site title
- Sidebar content
- Header content
- A page content title
- Default content

The easiest way to build skins for controls is to add the control to a page, set the control's properties, copy the generated ASP.NET markup code to the skin file, and delete the `ID` and `Text` properties. To create a skin for the Web site title, follow these steps:

1. **Drag and drop a label control onto a page on your Web site.**
2. **Press F4 to display the label's properties.**
3. **Set these properties:**
   - `Bold = True`
   - `Name = Verdana`
   - `Size = X-Large`
   - `ForeColor = Red`

   You can set the properties however you like. This book uses these values only as an example.
4. **Click the Source tab to display the page's source markup.**
5. **Copy the markup for the label you created and paste it in your skin file.**

   You add skin files the same way you add any new item: Right-click the `App_Themes` folder and choose Add New Item.
6. **Delete the `ID` and `Text` attributes from your label's markup.**

7. **Add the attribute `SkinID="siteTitle"` to the label's markup.**
8. **Add a comment above your label, like this:**

```
<%-- Web site title --%>
```

Repeat this process to add all your labels to the skin file.

To apply a skin to an individual control, set the control's `SkinID` property to the `SkinID` that you set in the skin file. For example, to apply the skin you create in the preceding set of steps to a label, follow these steps:

1. **Add a label control to your Web page.**
2. **Set the label's `SkinID` property to `siteTitle`.**

To test whether your skin is working, add text to your label and then press Ctrl+F5 to run your Web site.

A skin that uses a `SkinID` is a *named* skin. You can also define a default skin that applies to all controls of the same type by not using a `SkinID`. For example, the following skin applies to all label controls that use the theme with this skin:

```
<asp:Label runat="server" cssclass="label" />
```

TIP

You can use skins with CSS styles. For example, the following label control uses a CSS style. Any skins defined for the page are applied, also:

```
<asp:Label runat="server" cssclass="labelstyle" />
```

Mastering master pages

In the old days of development, developers used server-side includes and user controls to create a page template feel for their Web sites. Using the new ASP.NET master page feature, you no longer have to copy and paste repetitive code into your Web pages to create a template effect.

Although you could create page templates in previous versions of ASP.NET, you needed some coding expertise. The visual designer in Visual Studio didn't support custom page templates, either. Visual Studio has visual designer support for master pages.

The master page looks and feels like any Web page, with two major differences:

- ✦ Master pages aren't called from the browser.
- ✦ The `ContentPlaceHolder` control on the master page is replaced with content from a content page.

Think of the master page as the template for your Web site. Any content that you want repeated on each of your Web sites is added to this page. For example, add the header and footer and any navigation element to your master page. After you associate your content pages with your master page, those elements appear on all your content pages.

Visual Studio 2010 provides a default master page for you called `Site.master`.

To add another master page to an existing Web site:

1. **Right-click the Web site's project folder in Solution Explorer and choose Add New Item.**

   The Add New Item dialog box appears.

2. **Click the Master Page icon.**
3. **Click the Add button to add the master page to the Web site.**

   Visual Studio creates the master page and adds it to the Web site.

Adding content to your master page

Adding content to a master page is similar to adding content to any Web page. If you use table layout, add a table and start adding your header, footer, and other elements. If you use CSS layout, add `div` tags or other placeholders you like to use. See the upcoming section "Laying Out Web Pages with CSS" for more information on CSS layout.

You can also use ASP.NET expressions to add content to your master page. You can use expressions to add content to your page without typing the content in the page. For example, you can place information in a separate configuration file and then display that information on your master page. When you want to update the information, you only have to make a change in your configuration file.

Many people add configuration information to the appSettings section in the `web.config` file. To add copyright information to the AppSettings section of an existing Web site, follow these steps:

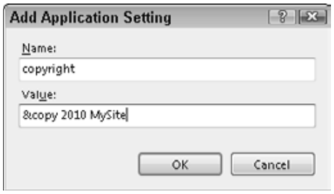
1. **Choose Administrative Tools➪Internet Information Services (IIS) Manager.**

   The Internet Information Services (IIS) Manager appears.

2. **Choose Application Settings from the ASP .NET area.**
3. **Right-click the list view and choose Add from the contextual menu.**

   The Add Application Setting dialog box, shown in Figure 6-6, appears.

4. **In the Name text box, type** copyright**.**
5. **In the Value text box, type** &copy 2010 MySite**.**



Name: copyright

Value: &copy 2010 MySite

Figure 6-6: Add application settings with the ASP.NET Web Site Administration tool.

6. **Click OK.**

   The name/value pair is saved in the `<appSettings>` section of the `web.config` file.
7. **Close your browser.**

Any time you want to change the site's copyright information, use the Web Site Administration Tool.

The syntax for an ASP.NET expression is

```
<%$ expressionPrefix: expressionValue %>
```

ASP.NET expressions are often used for managing connection strings to database resources. For more information on managing connection strings, see Book IV, Chapter 5.

To use your new copyright on your Web site, follow these steps:

1. **Drag and drop a `Literal` server control from the toolbox to your master page.**
2. **Set the control's `Text` property equal to `<%$ AppSettings: copyright %>`.**

Your copyright is displayed in Design view on the master page, as shown in Figure 6-7.

TIP

The `ContentPlaceHolder` control at the bottom of the screen, as shown in Figure 6-7, is where the content from your content pages appears. The control is added by default to the master page.

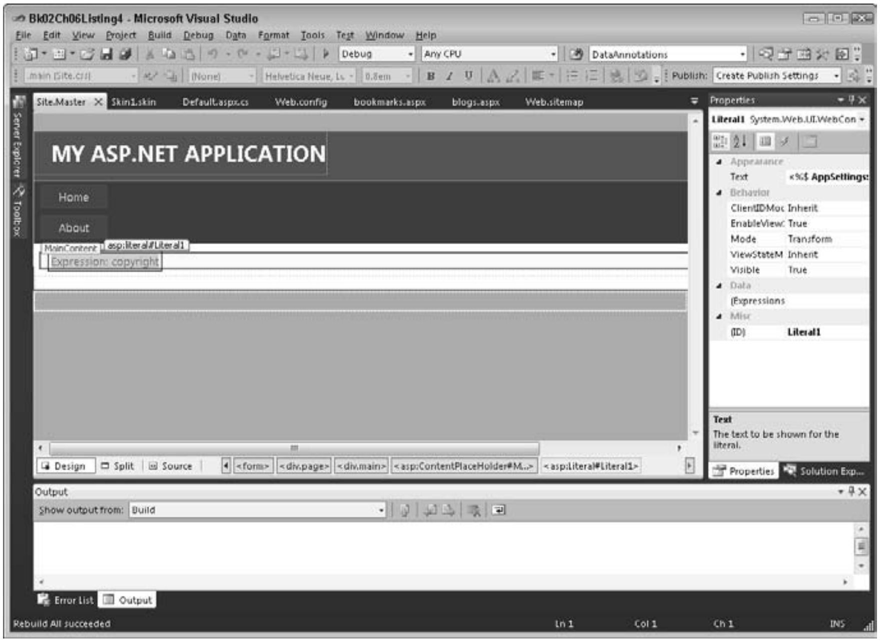


Figure 6-7: The copyright appears in Design view of the master page.

Adding new content pages

Without content from your Web pages, a master page is nothing more than a shell. You must add new Web pages to your site and associate them with the master page.

To add new content pages to use with your master page, follow these steps:

1. **Right-click the Web site's project folder in Solution Explorer and choose Add New Item from the contextual menu.**

   The Add New Item dialog box appears.

2. **Click the Web Form icon.**
3. **Enter a name for the Web form in the Name text box.**
4. **Select the Select Master Page check box.**
5. **Click Add.**

   The Select a Master Page dialog box appears.

6. **Click the master page that you want to associate with the Web form.**
7. **Click OK.**

The new Web form is different from a free-standing Web form in several ways:

- The attribute `MasterPageFile` is added to the `@Page` directive.
- A `ContentPlaceHolder` server control is added to the Web form.
- None of the usual HTML markup is present.

A page associated with a master page can't be executed on its own. As you add content to the page by using Design view, Visual Studio adds the content to the `ContentPlaceHolder` control on the page.

TIP

You can convert existing Web pages for use with master pages. Walk through the steps in this section so that you can see the differences between a Web page that doesn't use a master page and one that does. Then you can modify your Web page to make it work with the master page.

Accessing the master page

You can make your content pages and master pages talk to each other. For example, you can store the content page's title in the content page and then display the title on the master page.

To use the best-practices guideline, display a title for each page on the browser's title bar by using the HTML element title. The problem is that when you use master pages, the title element is on the master page. You need to display the title from the content page using the title element on the master page.

To set the master page's title element by using content from the content page, follow these steps:

1. **Add this script to the master page Source view between the `<head>` `</head>` tags:**

```
<script runat="server">
        string m_Title;

        public string Title
        {
            get
            {
                return m_Title;
            }
            set
            {
                m_Title = value;
            }
        }
        void Page_Load(Object o,EventArgs args)
        {
               if (!Page.IsPostBack)
               {
                  Mytitle.DataBind();
               }
            }
```

This C# code example creates a `Title` property for the master page. When the page loads, the master page's `DataBind` method is called on the `MyTitle` control, which tells the control to load data. For more information about data binding, see Book IV, Chapter 2.

2. **Add an `ID` attribute to the master page's title element:**

   ```
   <title id="Mytitle">
   ```

   The `ID` attribute makes it possible to access the title element with a script.

3. **Add the following ASP.NET expression between the opening and closing title tags:**

   ```
   <%# this.Title %>
   ```

   The title element appears this way:

   ```
   <title id="Mytitle">
     <%# this.Title %>
   </title>
   ```

   The expression evaluates to the `Title` property you create in Step 1.

4. **On the content page, add this directive below the `@Page` directive:**

   ```
   <%@ MasterType virtualpath="~/Site.Master" %>
   ```

   The `@ MasterType` directive creates a link between the content page and the master page so that you can access the master page's Title property.

5. **Add this script to the content page just below the `@ MasterType` directive you added in Step 4:**

   ```
   <script runat="server">
            void Page_Load(object o, EventArgs args)
            {
                Master.Title = "My Main Page";
            }

       </script>
   ```

   This C# code example sets the `Title` property of the master page to `My Main Page`.

6. **Run your site by pressing Ctrl+F5.**

   Your Web page displays the title on the browser's title bar, as shown in Figure 6-8.

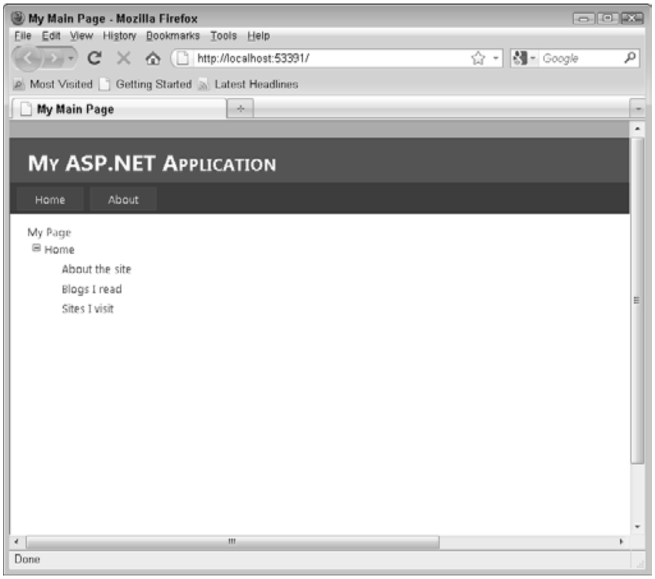


Figure 6-8: The title from the content page appears on the title bar.

You aren't limited to using the title in the title element. You can use it with any server control. For example, to display the title in a label control, you use the following code:

```
<asp:Label ID="TitleLabel" runat="server">
  <%# this.Title %>
</asp:Label>
```

Just remember to call the control's `DataBind` method. In this case, you'd call `TitleLabel.DataBind()`.

Laying Out Web Pages with CSS

You may be familiar with using CSS for setting fonts and background images. CSS also provides attributes for laying out and positioning elements on a Web page.

Regardless of whether you use master pages, you need a way to lay out your Web pages. Back when Web sites were first becoming popular, everyone used tables to lay out their Web pages. Using tables was a pain because of the markup that was involved. The World Wide Web Consortium (W3C) intended for tables to be used for displaying tabular data, not for laying out Web pages. Without an alternative, many designers used tables to lay out their Web pages, too.

Microsoft Web-design products have included visual designers for CSS layout for a few years now, and Microsoft is committed to supporting CSS layout and positioning in its Internet Explorer Web browser.

Laying out a Web page by using table-based layout requires tables and sometimes nested tables. The large amount of HTML markup required to create tables obscures your content. CSS layout eliminates the table markup tags in your content pages. Using CSS, you can store all your layout information in a file separately from your content. If you decide to change your layout or your content, you can look at each one in isolation. CSS is also a great way to address the accessibility issues. For example, you can replace the small font used on a Web site with a very large one to make it readable.

TIP

You can find several excellent free tutorials on the Internet for CSS. Two good choices are at `www.w3schools.com/css/default.asp` and `www.csstutorial.net`.

Introducing CSS basics

Visual Studio provides a style builder you can use for building CSS style rules. It helps if you're a little conversant in CSS syntax first. In a nutshell, CSS syntax is composed of these elements:

- **A selector:** An HTML element or custom-defined element that gets styled.
- **A declaration:** A set of attribute/value pairs.

Here are some examples of selectors:

- **HTML elements:** Examples are `h1`, `a`, and `input`.
- **Classes that can be applied to any element**: Classes use the syntax `.class`.
- **IDs that can be applied to any element:** Only one style ID can be used per page. IDs use the syntax `#id`.

Here are a couple of examples of declarations:

- Single-value declarations, such as `color: black;`
- Multiple-value declarations, such as `border: solid thick black;`

Put together a selector and some declarations, and you have a style rule that looks like this:

- Applies to instances of the HTML element `p`:

```
p
{
    color: Black;
    font: arial;
}
```

- Applies to all HTML elements with the attribute `class="container"`:

  ```
  .container
  {
      clear: both;
      border: solid thin red;
  }
  ```

- Applies to the single HTML element on the page, using the attribute `id="header"`:

  ```
  #header
  {
      position: relative;
      width: 798px;
  }
  ```

Here are some of the layout and positioning elements you should know about:

- **Top, Bottom:** Sets the top or bottom position of an element
- **Left, Right:** Sets the left or right position of an element
- **Height, Width:** Sets the height or width of an element
- **Z-index:** Sets the layer of an element where higher-order elements appear on top of lower-order elements
- **Position absolute:** Designates that an element is positioned according to the exact settings of the top, bottom, left, and right positions and the z-index
- **Position relative:** Designates that an element flows with the page layout and that any positioning attributes are relative to the flow of the page

Adding and creating CSS styles

You add CSS style rules to a style sheet. To add a style sheet to an existing Web site's theme folder, follow these steps:

1. **Right-click the Web site's `App_Themes` folder in Solution Explorer and then choose Add New Item from the contextual menu.**

   The Add New Item dialog box appears.

2. **Click the Style Sheet icon.**
3. **Click the Add button.**

   A style sheet is added to the folder.

To create styles, you first add a style and then build the style.

Adding a style

To create a style for all paragraphs on your Web site, follow these steps:

1. **Choose Styles➪Add Style Rule.**

   The Add Style Rule dialog box appears.

2. **From the Element drop-down list, select the HTML element for paragraph, P.**
3. **Click the arrow button in the center of the dialog box to move the `P` element into the Style Rule Hierarchy box on the right.**

   Use the style rule hierarchy to create styles that select elements nested within elements, classes, or element IDs.

4. **Click OK to add the style to the style sheet.**

Building a style

To build a style for the paragraph style, follow these steps:

1. **Choose Styles➪Build Style.**

   The Style Builder window appears.

2. **On the Font tab, click the drop-down arrow on the font-family drop-down list to select a font family, such as Arial.**
3. **Set additional attributes, such as font size and effects.**
4. **Click OK to build the style.**

   The style builder builds the style declaration in the style sheet for you.

Creating a page template with CSS

You can use the style builder to build the CSS layout for your Web page. To create a page template with a header, a left-side sidebar for navigation, a content placeholder, and a footer, use the style builder to create the styles. Follow these steps:

1. **Add style rules for each of these classes:**
   - `wrapper`
   - `header`
   - `leftnav`
   - `content`
   - `footer`

   Although making classes lowercase is common practice, it isn't required.

2. **Using the style builder, set the styles for each of the rules created in Step 1 so that they look like this:**

```
#wrapper
{
    width: 798px;
    height: 100%;
    text-align: left;
    margin: 0 auto 0 auto;
    border-left: solid thin black;
    border-right: solid thin black;
    border-bottom: solid thin black;
    background-color: White;
}
#header
{
    width: 100%;
    position: relative;
    height: 100px;
    border-bottom: solid thin black;
    background-color: #66CC33;
    vertical-align: middle;
}
#leftnav
{
    position: relative;
    float: left;
    width: 200px;
    border-right: groove thick #66CC33;
}
#content
{
    float: left;
    width: 70%;
    position: relative;
    margin-left: 25px;
}
#footer
{
    clear: both;
    width: 100%;
    background-color: #66CC33;
    border-top: solid thin black;
    font-size: smaller;
    padding-bottom: 10px;
}
```

3. **Add the following style to the `Body` element:**

```
text-align: center;
margin: 0;
background-color: #CCFF99;
font-family: Arial;
```

Use `div` tags to apply your layout classes to the master page Source view, as shown here:

```
<body>
    <form id="form1" runat="server">
        <div id="wrapper">
        <div id="header">
        </div>
        <div id="leftnav">
        </div>
        <div id="content">
            <asp:contentplaceholder id="mainContent" runat="server">
            </asp:contentplaceholder>
        </div>
        <div id="footer">
        </div>
        </div>
    </form>
</body>
```

Now, add content to your master page:

1. **Add a `TreeView` control in your `leftnav div` tags.**
2. **Add a `Breadcrumb` control to the `content div` tag before the content placeholder.**
3. **Add header information, such as your site name and site logo, in the `header div` tags.**
4. **Add your copyright information to the page footer.**

Learning to use CSS layout is no small feat. For more information, check out *CSS Web Design For Dummies,* by Richard Mansfield (Wiley). Microsoft provides several design template starter kits that use CSS layout. You can download starter kits for free at

```
http://msdn2.microsoft.com/en-us/asp.net/aa336613.aspx
```

To apply your style sheet to an existing Web page, follow these steps:

1. **Drag your style sheet from Solution Explorer.**
2. **Drop the style sheet on the Web page.**

Visual Studio adds a link tag to your Web page, to associate it with the style sheet.

Applying styles to controls

You have a few options for applying styles to controls. For example, you can create CSS classes to store styles. To apply a CSS class to a server control, follow these steps:

1. **Select the control on the Web page.**
2. **Press F4 to display the control's properties.**
3. **Add your CSS class to the `CssClass` property.**

You can also use the style builder to create styles for individual controls. To use the style builder with a control, follow these steps:

1. **Add a control to a Web page or select an existing control.**
2. **Right-click the control and choose Style from the contextual menu.**

   The Style Builder window appears.
3. **Build your style by using the style builder.**

   The style is applied to the control by using the control's `Style` property.

Chapter 7: Exploring ASP.NET Services

In This Chapter

- **Introducing ASP.NET 4 services**
- **Managing state in your Web site**
- **Creating a membership-enabled site**
- **Personalizing your Web site**

It has never been a better time to be an ASP.NET Web developer. After all those years of building tools for constructing Web sites, Microsoft finally realized that Web developers are trying to solve the same types of problems, such as registering people, logging in registered users, creating a personalized user experience, and monitoring the status of Web applications.

The latest version of ASP.NET provides the services developers need in order to build professional-quality Web sites without having to reinvent the wheel with each new Web site. This chapter introduces you to those new ASP.NET services and shows you how to use Visual Studio to leverage ASP.NET services in your next Web site project.

What a Wonderful Provider!

ASP.NET 4 offers many new services that provide common Web development features. These services include

- **Membership:** Creates and manages user accounts for providing access to a secure Web site
- **Role Manager:** Adds role-based management to your user accounts
- **Personalization:** Gives visitors to your site a personalized experience by storing user profile information or allowing them to create their own user experiences with Web parts
- **Session State:** Manages user sessions and chooses the data store that's appropriate for your situation
- **Site Navigation:** Shows your site visitors how to move around in your site by using site maps and navigation controls
- **Web Events:** Monitors and stores your Web site's health

As you can imagine, writing all the code yourself to provide these services would be time consuming. Each of these ASP.NET services provides a combination of Web server controls, public properties, and methods that implement the service, as well as access to a data store. And they all work well right out of the box.

ASP.NET services are made possible by the ASP.NET provider model. In this case, a *provider* is a type of software architecture that allows a framework, such as ASP.NET, to provide sophisticated services while hiding the implementation details of those services. The ASP.NET provider model has four layers:

- **Controls:** This set of Web server controls serves as the visible face of the service.
- **Services:** This public set of properties and methods, accessible by using Web server controls, provides the functionality of the service.
- **Providers:** All the implementation details that make the service "just work" happen by using these bits of code.
- **Data Stores:** This physical layer saves and retrieves the provider's data.

Table 7-1 shows the four layers of the ASP.NET provider model.

Table 7-1 Layers of the ASP.NET Provider Model

Layer	*Sample Technologies*
Controls	`Login`, `TreeView`
Services	Membership, site navigation
Providers	`SqlProvider`, `XMLProvider`, custom provider
Data Stores	SQL Server, XML document, custom data store

Why all these layers of abstraction? Why not just pass your data from the controls to the database? Creating distinct layers provides these benefits:

- You can customize any of the provider's layers without breaking the way the service works.
- The default implementation offers sophisticated services that work well out of the box.

Most ASP.NET services use an SQL Server database with a predefined database schema as the data store. If you decide that you want to change the data store's schema or use a completely different data store, you can create your own provider to access the data store. Any Web server controls that you to use keep working because the data access implementation details are hidden away in lower layers of the service.

For example, the Membership service uses an SQL Server database by default to store membership data. To use a new data store with the service, such as an XML file, you need to create a new provider. After you implement the details of accessing the XML data store in your new provider, telling the Membership service to use the new data store is a matter of changing the `SQLProvider` entry in the configuration file to `MyCustomXMLProvider`. Your membership-enabled Web site keeps working because the login controls are hidden from the details of the data store.

Using these ASP.NET services is a matter of using the controls associated with the service or setting configuration options in the Web site's `web.config` file.

A *`web.config`* file is an XML file that stores configuration settings for your Web site. To add a `web.config` file if your site doesn't have one yet, press Ctrl+Shift+A to open the Add New Item dialog box and select the Web Configuration File template.

Table 7-2 lists the ASP.NET services that use the provider model, and the control or the `web.config` section that enables each service.

Table 7-2 ASP.NET Services and Their Controls

Service	*Control or Type of Configuration Setting*
Membership	Login
Role management	`<roleManager>` section in `web.config`
Personalization	Web parts and `<profile>` section in `web.config`
Session state	Session object
Site navigation	Site map and navigation controls
Web events	`<healthMonitoring>` section in `web.config`

This chapter walks you through the process of using membership, role management, personalization, and session state services. Chapter 6 in this minibook covers site navigation.

If you're itching to get under the hood of the ASP.NET provider model so that you can create your own custom providers, check out the ASP.NET Provider Toolkit:

```
http://msdn.microsoft.com/en-us/aa336522.aspx
```

Managing State

Web applications are different from other kinds of applications, such as Windows applications, because Web pages use the HyperText Transfer Protocol (HTTP) for communications between the browser client and the server. HTTP is a *stateless* protocol: After a Web page is rendered to the browser, the Web server severs its connection to the client. As a result, the Web server has no memory of anything your visitor clicks on the Web page.

To get around this limitation of HTTP, Web frameworks such as ASP.NET provide many services for storing and retrieving a Web page's data after a user performs some action, such as clicking a button that requires processing on the server. Collectively, this process is *state management.*

You can use the ASP.NET state management features to develop your Web site without much consideration for how state is managed. ASP.NET takes care of all those details for you and hides their implementation. State management is one reason that frameworks such as ASP.NET are popular. Before ASP.NET, developers had to manage all state management details on their own. By using ASP.NET to handle the services of state management, you can focus on building your Web site.

The ASP.NET state-management offerings are client-side or server-side:

- **Client-side state management** stores state information on the client.
- **Server-side** stores state information on the server.

Generally, you should use client-side state management for storing small amounts of short-lived information. Server-side state management is more secure and can store more state information over longer time intervals than client-side options can.

Your client-side choices are described in this list:

- **View state:** A property used by ASP.NET Web pages
- **Hidden-field:** A standard HTML hidden field
- **Cookies:** Files stored on the client's computer for saving key/value pairs
- **Control State:** Custom control data stored between server trips
- **Query strings:** Key/value pairs appended to the end of a Web page URL

Here are the ASP.NET server-side offerings:

- **Application state:** Stores global values for the entire application to access
- **Session state:** Stores values for a user's session

- **Profile Properties**: Stores user-specific data
- **Database:** Stores state information in a custom database

Regardless of whether you choose client-side or server-side management, ASP.NET uses these state-management features by default:

- View state stores all values associated with the Web server controls on a page.
- Application and session state are stored in server memory.

Understanding view state

Every time you add a Web server control to a Web page, you're using view state to capture any data associated with that control. For example, if you add a `CheckBox` server control to a Web page, the user's check box selection is stored in view state. ASP.NET removes the user's check box from view state and sets the `checked` property on the check box equal to `True`. You don't have to know anything about view state to determine the check box's state. ASP.NET manages all that for you. You just query the check box's `checked` property.

You can use the view state to store your own data values independently of a Web server control. For example, here's how to add a customer's identification number to a Web page's view state:

1. **Add a text box control and a button control to a Web page.**
2. **In Design view of the page, double-click a blank area of the design surface.**

   Visual Studio creates a `Page_Load` event handler and opens the Code Editor.
3. **Save the customer's identification number to the view state by using these lines of code:**
   - **VB**

     ```
     If Not Page.IsPostBack Then
        ViewState("CustID") = "123456"
     End If
     ```
   - **C#**

     ```
     if( !Page.IsPostBack )
     {
        ViewState["CustID"] = "123456";
     }
     ```
4. **In Design view of the page, double-click the button control.**

   Visual Studio creates a `Button Click` event and opens the Code Editor.

5. **To retrieve the customer's identification number from view state and display it in the text box, type this line:**

   - **VB**

     ```
     me.TextBox1.Text = ViewState("CustID").ToString()
     ```

   - **C#**

     ```
     TextBox1.Text = ViewState["CustID"].ToString();
     ```

6. **Press Ctrl+F5 to run your Web site.**

When you first run your Web site, the text box is empty. After you click the button, the value `123456` appears in the text box. Here's how it works:

1. **When the page is first loaded in the browser, the Web page stores the value `123456` in its view state.**
2. **When you click the button, the browser sends the request to the Web server, along with the view state, and the Web server passes along the request with the view state data to ASP.NET.**
3. **ASP.NET picks apart the view state that the browser sent and provides state information to any controls.**

   ASP.NET holds on to the `123456` value.
4. **ASP.NET calls the `Page_Load` event. ASP.NET skips the code inside the `If` statement because the request is a postback.**

   A *postback* occurs when a request is sent from a Web page back to the Web server. See Book IV, Chapter 4 for more information about postbacks.
5. **The `Click` event is processed next, where `123456` is retrieved from the view state and placed in the text box.**

The view state is a key/value dictionary. It acts like you'd expect a dictionary to work. You add a key (such as `CustID`) and then a value (such as `123456`). You retrieve the value by using the key. Application state and session state are also key/value dictionaries. You add key/value pairs to application state and session state, just as you do to view state:

VB:

```
Object("Key")  = "value"
```

C#:

```
Object["Key"]  = "value";
```

Here's an example:

VB:

```
Application("AppStart") = Now
Session("CustID") = "123456"
```

C#:

```
Application["AppStart"]; = Now;
  Session["CustID"] = "123456";
```

Use view state whenever you need to

- **Store small amounts of data,** such as lookup keys for customer data or user identification data.
- **Pass data** from page to page and hide it from view.

Using session state

Whenever a user visits your Web site, a new session is created. ASP.NET creates a unique session identifier and associates it with the user's browsing session. You can store a collection of key/value pairs with the user's session. A collection of key/value pairs associated with a user's session is its *session state.*

Session state is enabled by default in ASP.NET. You can use session state to store information about a user's progress through your Web site on your Web server. Session state preserves the values across the lifetime of the user session as the user moves from page to page in your Web site.

Session state is stored in a dictionary and works similarly to view state. You can use session state anywhere you'd use view state. (See the preceding section for an example of how to use view state to store state information.)

Application state is similar to session state except that it stores values for the entire application, not just a single user's session.

Any time you need to store state, the view state, session state, and application state are at your disposal, with no additional configuration required.

Providing the Perks of Membership

One way to create a professional-looking Web site is to provide membership. *Membership* is the gateway to providing personalization features for your site, such as user profiles and Web parts. You can use membership to provide secure access to your Web site.

In the sections that follow, you find out how to set up and get started with membership.

Configuring the default database

Before you can configure your Web site to use membership, you need to configure the default application services provider database. You must have at a minimum SQL Server Express installed on your computer to set up the default provider database. After you do that, follow these steps:

1. **Launch the `Aspnet_regsql.exe` tool.**

   You can find this at

   ```
   drive:\WINDOWS\Microsoft.NET\Framework\version number
   ```

   The ASP.NET SQL Server Setup Wizard starts.

2. **Click Next to step through the wizard.**

   The Select a Setup Option page appears.

3. **On the Select a Setup Option page, select Configure SQL Server for Application Services and then click Next.**

   The Select the Server and Database page appears.

4. **Enter your database server's name and SQL Server instance.**

   Be sure to enter your SQL Server instance in the Server text box, as shown in Figure 7-1. The default instance name for SQL Server Express is SQLExpress. ***Note:*** You might have a name other than the default. The one used here is NEOPOLIS. You can get more information about SQL Server Express in Book IV, Chapter 6.

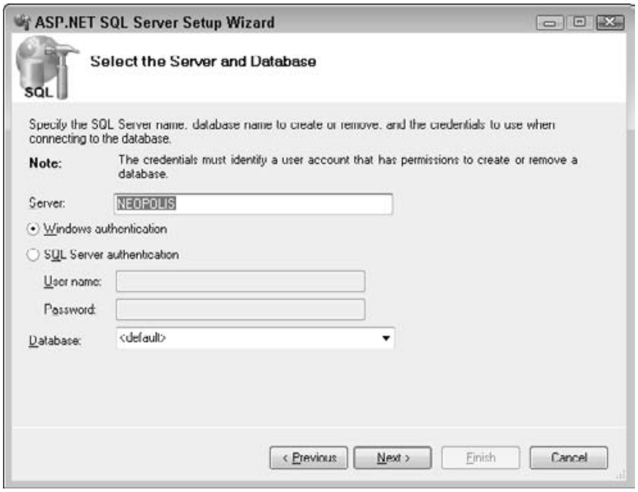
5. **Select the Windows Authentication radio button.**

6. **Accept the <default> database and click Next.**

7. **Click Next again to create the database.**

8. **Click Finish.**

You now have a default database that you can use for all ASP.NET services, including membership.



ASP.NET SQL Server Setup Wizard

Select the Server and Database

Specify the SQL Server name, database name to create or remove, and the credentials to use when connecting to the database.

Note: The credentials must identify a user account that has permissions to create or remove a database.

Server: NEOPOLIS

● Windows authentication

○ SQL Server authentication

User name:

Password:

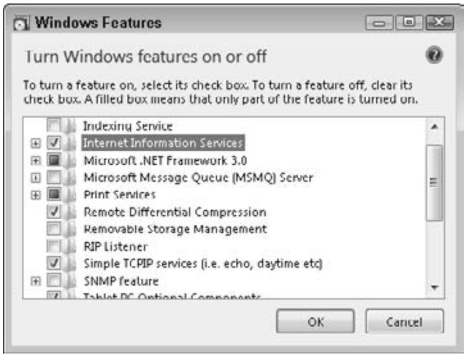
Database: <default>

Figure 7-1: Your settings are displayed on a summary page.

Running a site on IIS

You need to set up Internet Information Services (IIS) for the following examples to work. To do so, follow these steps:

1. **Open Control Panel.**
2. **Click the Programs applet.**
3. **Click the Turn Windows Features On or Off link.**
4. **Select Internet Information Services, as shown in Figure 7-2.**



Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- □ Indexing Service
- ☑ Internet Information Services
- ■ Microsoft .NET Framework 3.0
- □ Microsoft Message Queue (MSMQ) Server
- ■ Print Services
- ☑ Remote Differential Compression
- □ Removable Storage Management
- □ RIP Listener
- ☑ Simple TCPIP services (i.e. echo, daytime etc)
- □ SNMP feature
- ☑ Tablet PC Optional Components

Figure 7-2: Select Internet Information Services so that IIS will install.

You need a Web site running on an IIS server to use the ASP.NET membership features. To create a Web site running on IIS, follow these steps:

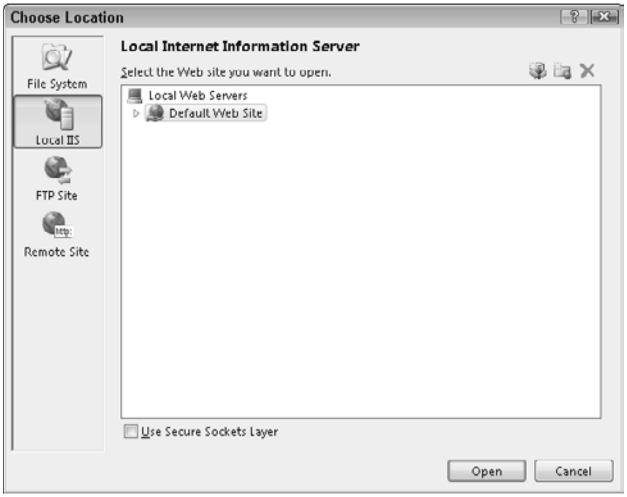
1. **Choose File➪New➪Web Site.**

   The New Web Site dialog box appears.

2. **Click the ASP.NET Web Site template.**
3. **Choose HTTP from the Location drop-down list.**
4. **Click the Browse button.**

   The Choose Location dialog box appears.

5. **Click Local IIS and then select Default Web Site from the site hierarchy, as shown in Figure 7-3, and click Open.**



Choose Location

Local Internet Information Server

Select the Web site you want to open.

Local Web Servers
Default Web Site

□ Use Secure Sockets Layer

Figure 7-3: A new Web site is added to the site hierarchy.

WARNING!

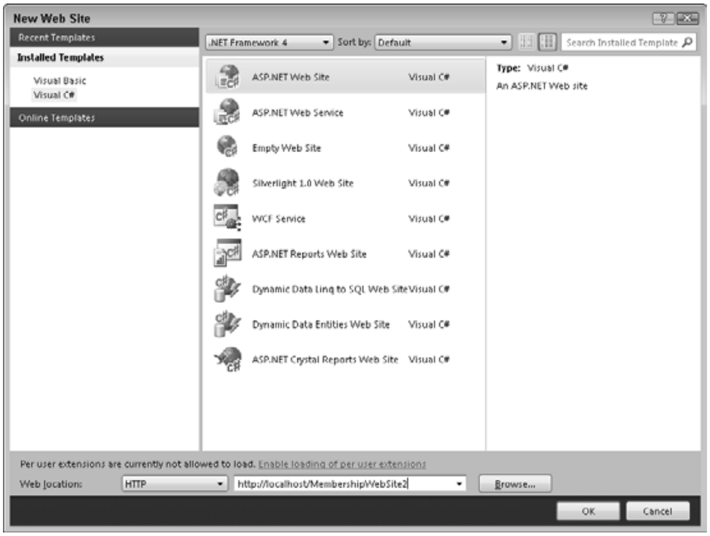
If you haven't installed IIS, when you click the Local IIS button, you may get an error message that reads

```
To access local IIS Web sites, you must install all of the following
    Windows components
```

If you get this error message, you need to install the required Windows components to access local IIS Web sites.

6. **Enter a name, such as MembershipWebSite2, in the Location field, as shown in Figure 7-4.**
7. **Click OK.**

   Visual Studio opens your new IIS Web site.


New Web Site
Recent Templates
Installed Templates
Visual Basic
Visual C#
Online Templates
Search Installed Template
ASP.NET Web Site Visual C#
ASP.NET Web Service Visual C#
Empty Web Site Visual C#
Silverlight 1.0 Web Site Visual C#
WCF Service Visual C#
ASP.NET Reports Web Site Visual C#
Dynamic Data Linq to SQL Web Site Visual C#
Dynamic Data Entities Web Site Visual C#
ASP.NET Crystal Reports Web Site Visual C#
Type: Visual C#
An ASP.NET Web site
Per user extensions are currently not allowed to load. Enable loading of per user extensions
Browse...
OK
Cancel

.NET Framework 4

Sort by: Default

Web location: HTTP http://localhost/MembershipWebSite2

Figure 7-4: Enter the location name here.

Securing membership content

The easiest way to secure content is when it's organized into folders. Add folders named Members and Administrative to the Web site.

To configure membership for the Web site, follow these steps:

1. **Choose Project⇨ASP.NET Configuration.**

   The ASP.NET Web Site Administration Tool is launched in a browser.

2. **Click the Security tab.**

   Use the Security tab to manage your users, roles, and access rules.

3. **On the Security tab, click the Use the Security Setup Wizard to Configure Security Step By Step link.**

   The Security Setup Wizard starts.

4. **Click Next to step through the wizard.**

   The Select Access Method appears.

5. **In Step 2 of the wizard, select the From the Internet option and then click Next.**

   Step 3 of the wizard confirms that you're using the default data store.

6. **Click Next to continue.**

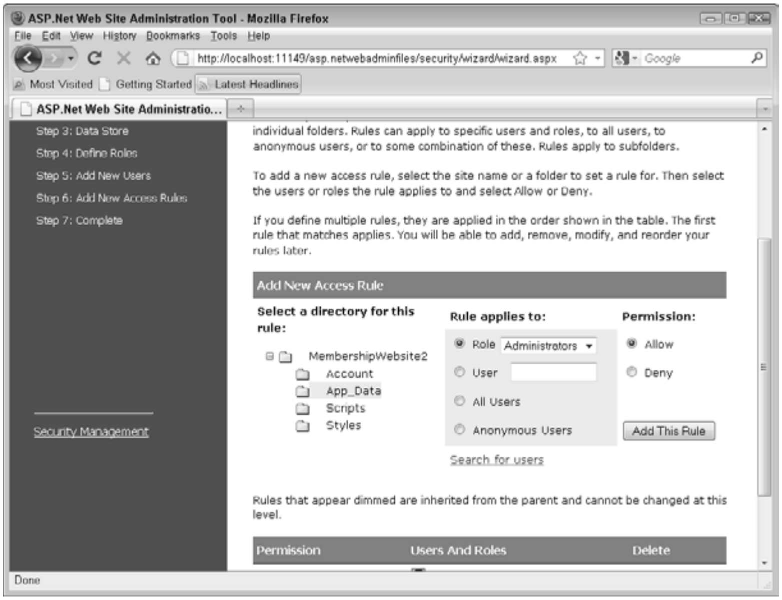
7. **To define roles for your members, select the Enable Roles for This Web Site check box and then click Next.**

8. **Type a new role —** Administrators **— and then click the Add Role button.**

9. **Repeat to add a Members role and click Next to continue.**

10. **In Step 5 of the wizard, create a user account for yourself and then click Next.**

    You see Step 6 of the wizard, shown in Figure 7-5.


ASP.Net Web Site Administration Tool - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:11149/asp.netwebadminfiles/security/wizard/wizard.aspx
Google
Most Visited
Getting Started
Latest Headlines
ASP.Net Web Site Administratio...
Step 3: Data Store
Step 4: Define Roles
Step 5: Add New Users
Step 6: Add New Access Rules
Step 7: Complete
Security Management
Add This Rule
Search for users
Done

individual folders. Rules can apply to specific users and roles, to all users, to anonymous users, or to some combination of these. Rules apply to subfolders.

To add a new access rule, select the site name or a folder to set a rule for. Then select the users or roles the rule applies to and select Allow or Deny.

If you define multiple rules, they are applied in the order shown in the table. The first rule that matches applies. You will be able to add, remove, modify, and reorder your rules later.

Add New Access Rule

Select a directory for this rule:

- MembershipWebsite2
  - Account
  - App_Data
  - Scripts
  - Styles

Rule applies to:

● Role Administrators
○ User
○ All Users
○ Anonymous Users

Permission:

● Allow
○ Deny

Rules that appear dimmed are inherited from the parent and cannot be changed at this level.

Permission	Users And Roles	Delete

Figure 7-5: Add an access rule for the Administrator role.

11. **Click the plus sign (+) next to the `MembershipWebsite2` directory hierarchy and then Select the App_Data folder.**

    The `MembershpWebsite2` directory hierarchy appears.

12. **In the Rule Applies To section, choose Administrators from the Role drop-down list.**

13. **Select the Allow radio button (from Permission) and then click the Add This Rule button.**

    The rule is added.

14. **With the Add New Access Rules page still displayed, select Anonymous Users in the Rule Applies to Section and select the Members folder in the `MembershipWebsite2` hierarchy.**

15. **Click the Deny radio button, click the Add This Rule button to add a rule denying anonymous users access to the Members folder, and then click Next.**

    The wizard is complete.

16. **Click the Finish button to close the wizard and return to the Security tab.**

Setting up a login page

You need a way for people to log into your Web site now that your site is set up for membership. Before ASP.NET 2.0, you had to create your own custom login pages. ASP.NET provides all the login controls you need in order to create a professional membership experience. The `Login` controls are shown in this list:

- `Login`: Provide the text boxes for username and password.
- `LoginView`: Create separate views for logged-in and anonymous users.
- `LoginStatus`: Tell users whether they're logged into the site.
- `LoginName`: Display a user's login name.
- `PasswordRecovery`: Retrieve user passwords by using their e-mail addresses.
- `CreateUserWizard`: Provide controls for self-registration.
- `ChangePassword`: Allow users to change their passwords.

`Login` controls are wired to work with the membership database. You don't have to write any code to make them work. The appearance of `Login` controls is customizable using control templates.

Using `Login` controls requires you to create a few Web pages. In your Web site, add these pages:

- `Login.aspx`: Your Web site's main login page must be named `Login.aspx` if you want to use the `LoginStatus` control.
- `Registration.aspx`: On this self-registration page, site visitors create their own usernames and passwords.
- `Members.aspx`: Add this page to your Members folder. A user must be logged into your site in order to access the page.

After you have the membership set up, create your Web site's start page for logging in:

1. **Add a `LoginStatus` control from the Login group of the toolbox to `default.aspx`, the start page of your Web page.**
2. **Add a `LoginView` control to your start page.**

3. **In the `LoginView` control AnonymousTemplate view, add a `Hyperlink` control from the Standard group in the toolbox.**

   The control has two view templates: Anonymous and Logged In. Access the view templates by clicking the arrow in the upper-right corner of the control.

4. **Press F4 to display the Properties window and then set these properties for the `Hyperlink` control:**

   - `Text`: Type **I Need To Register**.
   - `NavigateUrl`: Click the ellipsis and browse to `registration.aspx`.

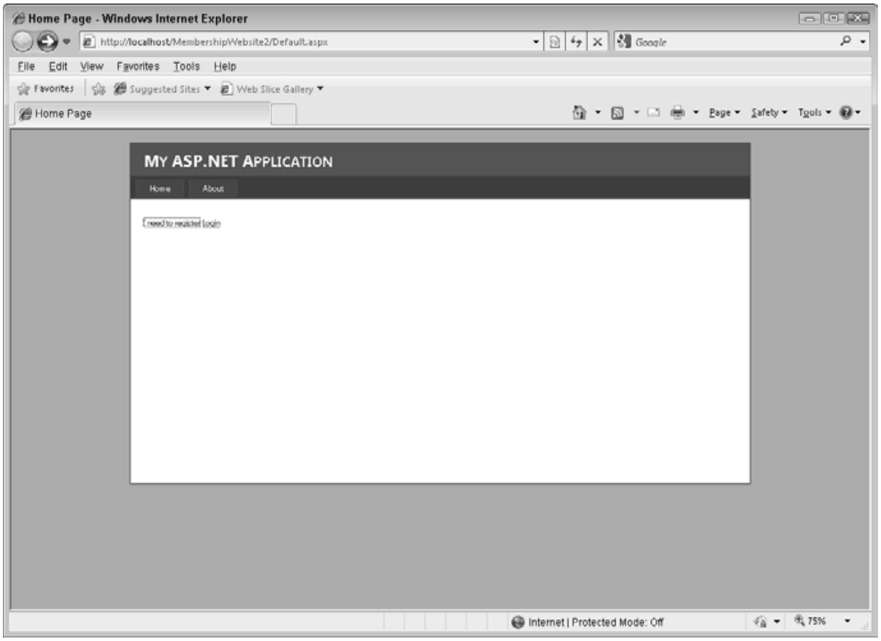
5. **In the `LoginView` control LoggedInTemplate view, type the text** Welcome, **.**

   Be sure to leave a space after the comma so that the user's name doesn't appear right next to the comma.

6. **Add the `LoginName` control to the `LoginView` control LoggedInTemplate immediately following the text you enter in Step 5.**

7. **Press Ctrl+F5 to run your site.**

   Your page displays Anonymous Login view, as shown in Figure 7-6.


Home Page - Windows Internet Explorer
http://localhost/MembershipWebsite2/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page Safety Tools
Internet | Protected Mode: Off
75%

My ASP.NET Application

Home About

I need to register Login

Figure 7-6: This page displays the `Login View` anonymous template.

Now, you need to add login and user-registration capabilities to your site. To set up the login page, `Login.aspx`, follow these steps:

1. **Add a `Login` control from the Login group of the toolbox.**
2. **Set the control's appearance by using the Auto Format feature.**
3. **Set the control's `DestinationPageUrl` property to the `default.aspx` page in the Members folder.**

To set up the registration page, `Registration.aspx`, follow these steps:

1. **Add the `CreateUserWizard` control from the toolbox's Login group.**
2. **Set the control's `ContinueDestinationPageURL` property to the `default.aspx` page in the Members folder.**

Run your Web site and test it. You should be able to log in and create a new user account.

You can use membership and login controls in conjunction with master pages to keep the look and feel of your login page similar to the rest of your site. (Turn to Book IV, Chapter 6 for more on master pages.) Figure 7-7 shows you a master page using login controls in the Web site you create in Book IV, Chapter 6. Users who aren't logged in are directed to the login page.

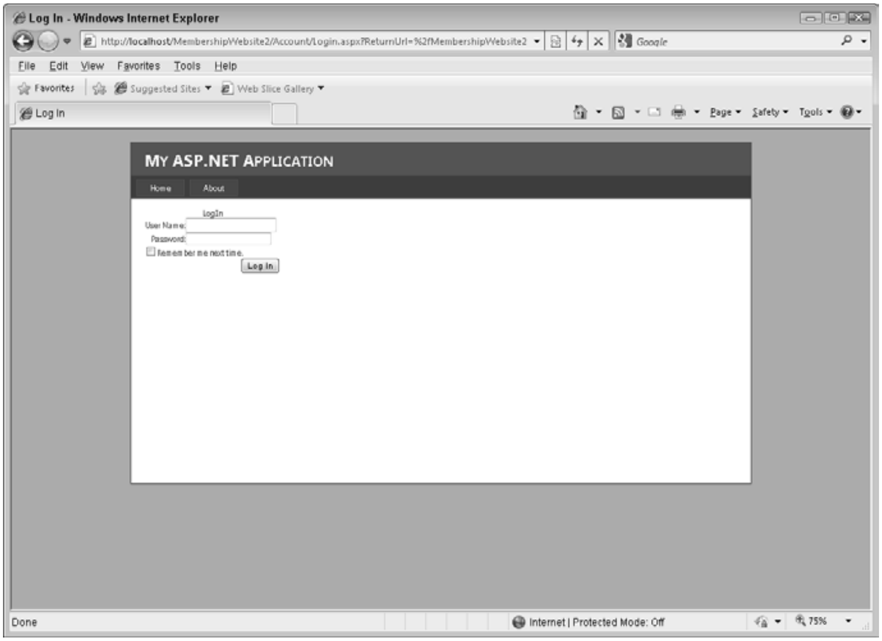

Log In - Windows Internet Explorer
http://localhost/MembershipWebsite2/Account/Login.aspx?ReturnUrl=%2fMembershipWebsite2
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Log In
Page Safety Tools
MY ASP.NET APPLICATION
Home
About
Log In
User Name:
Password:
Remember me next time.
Log In
Done
Internet | Protected Mode: Off
75%

Figure 7-7: Membership and login controls go hand in hand with master pages and site navigation.

Getting Personal

Even the simplest Web site now offers personalization features that allow a visitor to create a personalized user experience. This section introduces two ASP.NET personalization features: profiles and Web parts.

Saving visitor settings with profiles

A *profile* is used to store and retrieve user settings from a data source, such as a database. Examples of profile attributes include

- Contact information
- A favorite movie
- Preferences, such as themes

The ASP.NET profile service is easy to configure and use. It requires the application services default data store or a custom data store of your choosing. (Read the earlier section "Providing the Perks of Membership" to see how to create the default data store.) If you already created the data store, the profile service uses it.

To enable profiles for your Web site, you must add the profile attributes to your Web site's `web.config` file. To add a favorite color attribute to your profile, follow these steps:

1. **Open the `web.config` file.**

   A *`web.config`* file is an XML file that stores configuration settings for your Web site. To add a `web.config` file if your site doesn't have one yet, press Ctrl+Shift+A to open the Add New Item dialog box and select the Web Configuration File template.

2. **Between the `system.web` open and closing tags, add this code:**

```
<profile>
  <properties>
    <add name="FavoriteColor"
      type="System.String"  />
  </properties>
</profile>
```

It's that easy! You can start saving the favorite colors of all your logged-in users.

By default, only logged-in users can use properties. To allow anonymous users to use properties, follow these steps:

1. **Add this line to the `web.config` file:**

```
<anonymousIdentification enabled="true" />
```

2. **Add this line to the Properties section:**

   ```
   <name="Property" allowAnonymous="true" />
   ```

To use the profile in your Web site, follow these steps:

1. **Add a `profiles.aspx` page to your Web site.**
2. **Add `Label`, `Textbox`, and `Button` server controls to the page.**
3. **Set the label's `Text` property to `Favorite Color`.**
4. **Double-click the button to add code to its `Click` event.**

   The Code Editor appears.
5. **Type this line to set the profile attribute:**

   ```
   Profile.FavoriteColor = Me.TextBox1.Text
   ```
6. **Double-click Design view of the page to access the page's `Load` event.**

   The Code Editor appears.
7. **Type this line to load the profile's property:**

   ```
   If Not Page.IsPostBack Then
       Me.TextBox1.Text = Profile.FavoriteColor
   End If
   ```

To test your profile, run your site and log in. Enter a favorite color and click the button to save the profile.

Letting users customize with Web parts

Web parts controls are the ultimate in personalization. What makes them powerful is that they allow users to customize their user experience. A visitor accessing a Web parts page can add and remove Web parts or change the layout of the page.

Web parts controls have three elements:

- **User interface (UI) controls:** These controls are the standard ASP.NET server controls, such as labels, text boxes, and buttons.
- **UI structure components:** Structural components provide the layout and management features of Web parts. The two main structural components are the Web parts manager and the Web part zone.
- **Personalization layer:** This underlying infrastructure layer allows users to change the page layout by moving controls around, or adding and removing controls from the page. Users' changes are saved permanently.

The Web parts controls are accessible from the WebParts group in the toolbox. You should be familiar with these Web parts:

- `WebPartManager`: Required by each page in order to manage the Web parts on the page.
- `WebPartZone`: Provides zones for laying out Web parts controls. A page can have multiple zones.
- `CatalogZone`: Provides a zone for creating a catalog of Web parts that a user can use to modify the page.
- `EditorZone`: Allows users to edit the Web parts on a page.
- `ConnectionsZone`: Enables you to create connections among Web parts.

Note that the zone-related controls are logical zones for working with Web parts. You're still responsible for laying out the page by using tables or Cascading Style Sheets (CSS).

REMEMBER

The main reason for using Web parts is their personalization features. Getting personalization up and running is no small feat. Use Web parts and Web parts pages for building portals that allow visitors to customize their views. A popular portal that uses Web parts is Microsoft SharePoint Portal Server 2003. SharePoint Portal Server's underlying infrastructure, Windows SharePoint Services, also uses Web parts.

Chapter 8: Using Threads and Timers

In This Chapter

- **Dividing your work with threads and thread pools**
- **Queuing up your work using the .NET framework thread pool**
- **Keeping things on schedule with timers**

In the early days of software, applications were pretty simple. Computers could typically run one application at a time so if you were typing a document and wanted to update items on a spreadsheet, you had to quit your word-processing application and then launch your spreadsheet application. Software applications were also pretty simple. They just waited for you to enter a command, and then they'd execute the command and return to wait for you to enter another command.

Today, users demand much more from their software. Users expect software to be responsive no matter what. If you click a button to add an item to a database in a faraway location, users expect to move the window or perform some other task while the database request is still processing. Also, gone are the days where getting updates to your software required sending a request to the manufacturer and getting a set of disks in the mail with the latest version of the software. Today, most software applications connect to the network and periodically check for updates. If the manufacturer has updates to your software, the software itself retrieves them across the network and updates the application, even while you're using it.

How is it that a software application can send a request to update a database in a faraway location, download updates from the manufacturer, and still respond to input from the user to process more requests? Threads and thread pools, which I cover in this chapter, make all these events happen. In this chapter, you discover how to use threads in your applications and also how to take advantage of the built-in .NET thread pool.

Dividing Up Your Work with Threads and Thread Pools

Threads are simply separate units of execution that a processor can schedule. An application may have one thread that responds to user inputs and redraws the window, another that executes requests to a database, and yet another that downloads updates from the manufacturer. The processor can schedule each thread so that they don't block each other from executing.

As a result, when it comes time to send a request to the database, the processor schedules that job to execute and lets the other thread still respond to input from the user. If the user clicks a button, the processor simply schedules another task to respond without blocking the database operation. The processor shares execution time among all threads in the application so it appears to the user that many things are happening at the same time. In fact, with modern multiprocessor architectures, many threads can now be executing simultaneously, each on a separate processor. Chapter 10 of Book IV explains more about parallel programming on multiprocessor architectures.

Starting and stopping threads is a very processor-intensive operation. Also, creating more threads than your application needs isn't an efficient use of processor resources. Ideally, you want your application to create only the threads it needs to efficiently perform its work and no more. To solve this problem, software developers came up with the idea of *thread pools.*

Simply put, thread pools are a set of threads that are created to handle tasks. Threads in a thread pool monitor queues that contain jobs to execute. Threads in a thread pool are in a waiting state waiting for jobs to put on the queue. When jobs are put in the queue, threads that aren't already executing a job pull jobs off of the queue, execute them, and then go back to waiting when their work is finished. More sophisticated thread pools keep track of the number of jobs in the queue and can create and destroy threads as the amount of work changes. This technique is used in many modern software applications, but designing and creating a thread pool for each application you want to write is tedious.

Fortunately, the .NET framework has eliminated some of the work by implementing a sophisticated thread pool that is easy for the programmer to use. Threading in the .NET framework is provided in the `System.Threading` namespace in the `System.Core.dll` assembly.

Adding Threads to Your Application

The .NET framework makes it simple to add threading to your application. The following console application illustrates threading in .NET.

C#:

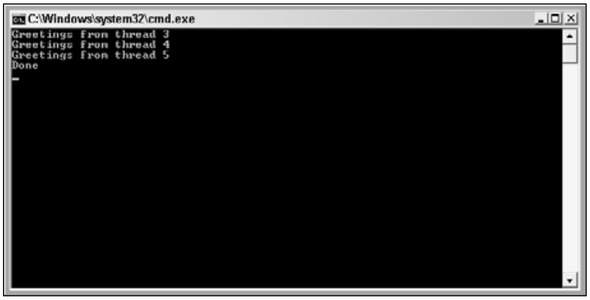
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
```

```
namespace SimpleThreadCS
{
    class Program
    {
        public static void ThreadFunction()
        {
            System.Console.WriteLine("Greetings from thread {0}.", Thread.
    CurrentThread.ManagedThreadId);
        }

        static void Main(string[] args)
        {
            Thread t1 = new Thread(new ThreadStart(ThreadFunction));
            Thread t2 = new Thread(new ThreadStart(ThreadFunction));
            Thread t3 = new Thread(new ThreadStart(ThreadFunction));

            t1.Start();
            t2.Start();
            t3.Start();

            /*Wait for all threads to finish*/
            t1.Join();
            t2.Join();
            t3.Join();

            System.Console.WriteLine("Done");
            System.Console.ReadLine();
        }
    }
}
```

VB:

```
Imports System
Imports System.Threading

Module Module1

    Public Sub ThreadFunction()
        System.Console.WriteLine("Greetings from thread {0}", Thread.
    CurrentThread.ManagedThreadId)

    End Sub
    Sub Main()
        Dim t1 As Thread = New Thread(New ThreadStart(AddressOf ThreadFunction))
        Dim t2 As Thread = New Thread(New ThreadStart(AddressOf ThreadFunction))
        Dim t3 As Thread = New Thread(New ThreadStart(AddressOf ThreadFunction))

        t1.Start()
        t2.Start()
        t3.Start()

        t1.Join()
        t2.Join()
        t3.Join()

        System.Console.WriteLine("Done")
        System.Console.ReadLine()

    End Sub

End Module
```

This example creates three new threads that all execute ThreadFunction. ThreadFunction writes a message containing its thread identifier to the console. The thread that creates the three threads waits for all to finish and then writes Done to the console as shown in Figure 8-1.



```
Greetings from thread 3
Greetings from thread 4
Greetings from thread 5
Done
_
```

Figure 8-1: A simple thread application.

Notice that no data is provided to the threads in ThreadFunction. Often, you want multiple threads to do the same operation, but perhaps on different data. In this case, providing a parameter to ThreadFunction that contains data that you want the thread to process is a good idea. Modifying the preceding example slightly to create an object containing data to provide to ThreadFunction, the example now looks as follows:

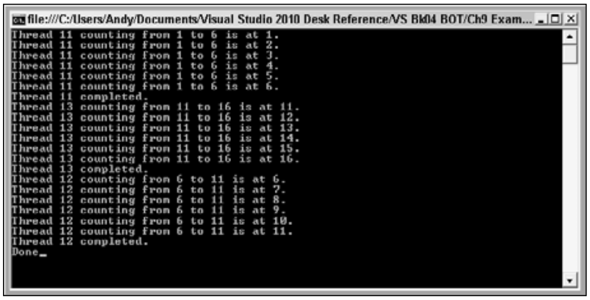
C#:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;

namespace SimpleThread2CS
{
    class MyStruct
    {
        public MyStruct(int x, int y)
        {
            start = x;
            count = y;
        }
        public int start;
        public int count;
    };

    class Program
    {

        static void ThreadFunction(object o)
        {
            MyStruct s = o as MyStruct;

            for (int i = s.start; i <= (s.start + s.count); ++i)
            {
                Console.WriteLine("Thread {0} counting from {1} to {2} is at
    {3}.",
                                   Thread.CurrentThread.ManagedThreadId, s.start,
    s.start + s.count, i);
            }

            Console.WriteLine("Thread {0} completed.", Thread.CurrentThread.
    ManagedThreadId);
        }

        static void Main(string[] args)
        {
            Thread t1 = new Thread(new ParameterizedThreadStart(ThreadFunction));
            Thread t2 = new Thread(new ParameterizedThreadStart(ThreadFunction));
            Thread t3 = new Thread(new ParameterizedThreadStart(ThreadFunction));

            t1.Start(new MyStruct(1, 5));
            t2.Start(new MyStruct(6, 5));
            t3.Start(new MyStruct(11, 5));

            t1.Join();
            t2.Join();
            t3.Join();

            Console.WriteLine("Done");
            Console.ReadLine();

        }
    }
}
```

VB:

```
Imports System
Imports System.Threading

Module Module1

    Class MyStruct

        Public Sub New(ByVal x As Integer, ByVal y As Integer)
            start = x
            count = y
        End Sub

        Public count As Integer
        Public start As Integer


    End Class

    Sub ThreadFunction(ByVal o As Object)
        Dim s As MyStruct = o

        For i As Integer = s.start To (s.start + s.count) Step 1
            Console.WriteLine("Thread {0} counting from {1} to {2} is at {3}.",
                             Thread.CurrentThread.ManagedThreadId, s.start,
    s.start + s.count, i)

        Next i
        Console.WriteLine("Thread {0} completed.", Thread.CurrentThread.
    ManagedThreadId)

    End Sub
    Sub Main()
        Dim t1 As Thread = New Thread(New ParameterizedThreadStart(AddressOf
    ThreadFunction))
        Dim t2 As Thread = New Thread(New ParameterizedThreadStart(AddressOf
    ThreadFunction))
        Dim t3 As Thread = New Thread(New ParameterizedThreadStart(AddressOf
    ThreadFunction))

        t1.Start(New MyStruct(1, 5))
        t2.Start(New MyStruct(6, 5))
        t3.Start(New MyStruct(11, 5))

        t1.Join()
        t2.Join()
        t3.Join()

        Console.Write("Done")
        Console.ReadLine()

    End Sub

End Module
```

The output appears in Figure 8-2.



file:///C:/Users/Andy/Documents/Visual Studio 2010 Desk Reference/VS Bk04 BOT/Ch9 Exam...

```
Thread 11 counting from 1 to 6 is at 1.
Thread 11 counting from 1 to 6 is at 2.
Thread 11 counting from 1 to 6 is at 3.
Thread 11 counting from 1 to 6 is at 4.
Thread 11 counting from 1 to 6 is at 5.
Thread 11 counting from 1 to 6 is at 6.
Thread 11 completed.
Thread 13 counting from 11 to 16 is at 11.
Thread 13 counting from 11 to 16 is at 12.
Thread 13 counting from 11 to 16 is at 13.
Thread 13 counting from 11 to 16 is at 14.
Thread 13 counting from 11 to 16 is at 15.
Thread 13 counting from 11 to 16 is at 16.
Thread 13 completed.
Thread 12 counting from 6 to 11 is at 6.
Thread 12 counting from 6 to 11 is at 7.
Thread 12 counting from 6 to 11 is at 8.
Thread 12 counting from 6 to 11 is at 9.
Thread 12 counting from 6 to 11 is at 10.
Thread 12 counting from 6 to 11 is at 11.
Thread 12 completed.
Done_
```

Figure 8-2: A simple thread application with data.

The preceding example is more useful than the first example. Now each thread can perform the same operation, but on different data. It is still limited, however, because the only data type that `ThreadFunction` can process is `MyStruct`. If you have a function that requires more than one parameter, you're out of luck. In the preceding case, you have to create a class containing the data you want to pass to the thread, which isn't always convenient.

Queuing Your Work Using the .NET Framework Thread Pool

Creating threads in the .NET framework is fairly simple, but many times in your application, you don't know how many threads you'll need. Also, by using the `System.Threading.Thread` class, you're responsible for creating and destroying threads as well as managing the lifetime of the threads.

Fortunately, the .NET framework provides a built-in thread pool for you. Instead of worrying about the details of thread management, the thread pool does it for you, and you can just concentrate on creating the classes and methods you need to meet the requirements of your application. The built-in thread pool is available in the `System.Threading` namespace.

Delegating responsibility using delegates

Delegates have existed in the .NET framework since the beginning. Delegates are a special type of class that you can assign methods to. You can pass delegates to methods and make them data members of a class. Delegates provide a convenient model for executing methods either synchronously or asynchronously in another thread. Delegates can have any number of arguments as long as the delegate definition matches the signature of the method assigned to it.

You can modify the previous examples to have threads execute methods with a variable number of arguments. Modifying the previous example, you can see how delegates make it easy to have different threads execute your methods in your application. Also, notice that there are no threads to create. The .NET framework takes care of this for you.

When you call `BeginInvoke` on a delegate, the framework places this delegate on the job queue of the built-in thread pool and executes it on the next available thread:

C#:

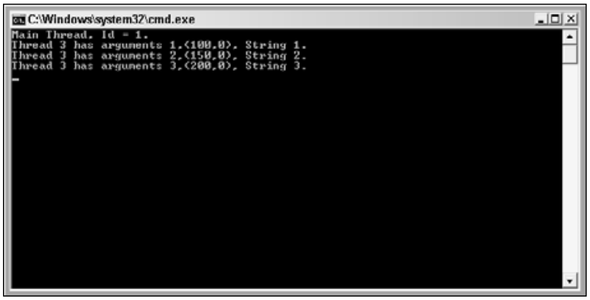
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;

namespace ThreadsWithDelegatesCS
{
    class MyStruct
    {
        public MyStruct(int x, int y)
        {
            start = x;
            finish = y;
        }

        public override string ToString()
        {
            return ("(" + start + "," + finish + ")");
        }

        public int start;
        public int finish;
    };

    class Program
    {
        private delegate void ThreadFunctionDelegate(int arg1, MyStruct arg2,
    string arg3);

        static void ThreadFunction(int arg1, MyStruct arg2, string arg3)
        {
            System.Console.WriteLine("Thread {0} has arguments {1},{2}, {3}.",
                                     Thread.CurrentThread.ManagedThreadId,arg1,
    arg2.ToString(), arg3);
        }

        static void Main(string[] args)
        {
            ThreadFunctionDelegate d1 = new ThreadFunctionDelegate(Thread
    Function);
            ThreadFunctionDelegate d2 = new ThreadFunctionDelegate(Thread
    Function);
            ThreadFunctionDelegate d3 = new ThreadFunctionDelegate(Thread
    Function);

            Console.WriteLine("Main Thread, Id = {0}.", Thread.CurrentThread.
    ManagedThreadId);

            d1.BeginInvoke(1, new MyStruct(50, 100), "String 1", null, null);
            d2.BeginInvoke(2, new MyStruct(100, 150), "String 2", null, null);
            d3.BeginInvoke(3, new MyStruct(150, 200), "String 3", null, null);

            Thread.SpinWait(10000000);

            Console.ReadLine();

            Console.WriteLine("Done");

        }
    }
}
```

VB:

```
Imports System
Imports System.Threading

Module Module1

    Class MyStruct
        Public start As Integer
        Public finish As Integer
        Public Sub New(ByVal x As Integer, ByVal y As Integer)
            start = x
            start = y
        End Sub

        Public Overrides Function ToString() As String
            Return ("(" & start & "," & finish & ")")
        End Function

    End Class

    Delegate Sub ThreadFunctionDelegate(ByVal arg1 As Integer, ByRef arg2 As
    MyStruct, ByVal arg3 As String)

    Sub ThreadFunction(ByVal arg1 As Integer, ByRef arg2 As MyStruct, ByVal arg3
    As String)
        System.Console.WriteLine("Thread {0} has arguments {1},{2}, {3}.",
                                Thread.CurrentThread.ManagedThreadId, arg1, arg2.
    ToString(), arg3)

    End Sub

    Sub Main()
        Dim d1 As ThreadFunctionDelegate = New ThreadFunctionDelegate(AddressOf
    ThreadFunction)
        Dim d2 As ThreadFunctionDelegate = New ThreadFunctionDelegate(AddressOf
    ThreadFunction)
        Dim d3 As ThreadFunctionDelegate = New ThreadFunctionDelegate(AddressOf
    ThreadFunction)

        Console.WriteLine("Main Thread, Id = {0}.", Thread.CurrentThread.
    ManagedThreadId)

        d1.BeginInvoke(1, New MyStruct(50, 100), "String 1", Nothing, Nothing)
        d2.BeginInvoke(2, New MyStruct(100, 150), "String 2", Nothing, Nothing)
        d3.BeginInvoke(3, New MyStruct(150, 200), "String 3", Nothing, Nothing)

        Thread.SpinWait(10000000)

        Console.ReadLine()

        Console.WriteLine("Done")

    End Sub

End Module
```

In the preceding example, calling BeginInvoke on d1, d2, and d3 handles all the details of creating a job on the thread pool and passing the arguments to ThreadFunction so that it can execute on a separate thread. Instead of creating a separate thread for each of the three invocations of ThreadFunction, in my case, the thread pool happened to use the same thread to execute ThreadFunction all three times, as shown in Figure 8-3.

In the preceding example, you create a delegate type for any type of method you may have and have the thread pool execute it asynchronously on a worker thread for you, freeing up the calling thread to do something else.



```
Main Thread. Id = 1.
Thread 3 has arguments 1,(100,0), String 1.
Thread 3 has arguments 2,(150,0), String 2.
Thread 3 has arguments 3,(200,0), String 3.
_
```

Figure 8-3: Threading with delegates.

One limitation still exists, however. More often than not, you have functions that return data. So what happens if you want to execute a method asynchronously on the thread pool and want to return data from this method? So far, you haven't seen any examples that do this task, but in the following example, you see how to execute methods that return data asynchronously.

Returning data from delegates

The `BeginInvoke` method on the delegate class has a method signature:

C#:

```
public IAsyncResult BeginInvoke(Delegate method, params Object[] args,
    AsyncCallback cb, object state);
```

VB:

```
Public Function BeginInvoke(method As Delegate, ParamArray args as Object(), cb
    As AsyncCallback, state As Object) As IAsyncResult
```

The `IAsyncResult` return value from `BeginInvoke` and the last two parameters of `BeginInvoke` make returning data from delegates happen. `AsyncCallback` is a special type of delegate that allows you to provide a method that the thread pool will call when the method invoked by `BeginInvoke` completes. The last argument allows you to provide an object that will be provided to your `AsyncCallback` method when the delegate finishes executing. A method provided to `AsyncCallback` has the following signature:

C#:

```
public void OnActionCompleted(IAsyncResult result);
```

VB:

```
Public Sub OnActionCompleted(result As IAsyncResult)
```

The delegate class has another handy method called EndInvoke, which has the following signature:

C#:

```
public Object EndInvoke(IAsyncResult result);
```

VB:

```
Public Function EndInvoke(result as IAsyncResult) As Object
```

Here is where the magic happens. The method you provided to BeginInvoke in the AsyncCallback parameter gets called when the delegate finishes executing, and you get the IAsyncResult object returned by BeginInvoke as an argument to your callback function. Remember the last argument to BeginInvoke? This argument is an object that the framework will pass to your callback method when the delegate completes. The delegate class has an EndInvoke method that returns data of type Object. It just so happens the return value contains the same data that the method executed by the delegate returns. So if you pass the delegate object itself as the last argument to BeginInvoke, the framework will give you back this object in your callback method. Now, you can call EndInvoke on that delegate and get your data from the asynchronous method call. The following example illustrates returning data from delegates executed on the thread pool.

C#:

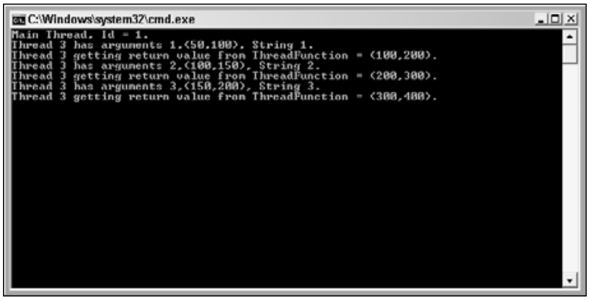
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;

namespace DelgatesReturnDataCS
{
    class MyStruct
    {
        public MyStruct(int x, int y)
        {
            start = x;
            finish = y;
        }

        public override string ToString()
        {
            return ("(" + start + "," + finish + ")");
        }

        public int start;
        public int finish;
    };

    class MyReturnStruct
    {
        public MyReturnStruct(int x, int y)
    {
        xValue = x;
        yValue = y;
    }

    public int xValue;
    public int yValue;

    public override string ToString()
    {
        return ("(" + xValue + "," + yValue
            + ")");
    }

};

class Program
{
    private delegate MyReturnStruct ThreadFunctionDelegate(int arg1, MyStruct
arg2, string arg3);

    static MyReturnStruct ThreadFunction(int arg1, MyStruct arg2, string
arg3)
    {
        System.Console.WriteLine("Thread {0} has arguments {1},{2}, {3}.",
                                 Thread.CurrentThread.ManagedThreadId, arg1,
arg2.ToString(), arg3);

        return (new MyReturnStruct(arg2.start * 2, arg2.finish * 2));
    }

    static void OnThreadFunctionComplete(IAsyncResult result)
    {
        ThreadFunctionDelegate del = result.AsyncState as
ThreadFunctionDelegate;

        MyReturnStruct val = del.EndInvoke(result);

        Console.WriteLine("Thread {0} getting return value from
ThreadFunction = {1}.",
                          Thread.CurrentThread.ManagedThreadId, val.
ToString());
    }

    static void Main(string[] args)
    {
        ThreadFunctionDelegate d1 = new ThreadFunctionDelegate(Thread
Function);
        ThreadFunctionDelegate d2 = new ThreadFunctionDelegate(Thread
Function);
        ThreadFunctionDelegate d3 = new ThreadFunctionDelegate(Thread
Function);

        Console.WriteLine("Main Thread, Id = {0}.", Thread.CurrentThread.
ManagedThreadId);

        d1.BeginInvoke(1, new MyStruct(50, 100), "String 1", new AsyncCall
back(OnThreadFunctionComplete), d1);
        d2.BeginInvoke(2, new MyStruct(100, 150), "String 2", new AsyncCall
back(OnThreadFunctionComplete), d2);
        d3.BeginInvoke(3, new MyStruct(150, 200), "String 3", new AsyncCall
back(OnThreadFunctionComplete), d3);
```

```
            Thread.SpinWait(10000000);

            Console.ReadLine();

            Console.WriteLine("Done");
        }
    }
}
```

VB:

```
Imports System
Imports System.Threading

Module Module1

    Class MyStruct
        Public start As Integer
        Public finish As Integer

        Public Sub New(ByVal x As Integer, ByVal y As Integer)
            start = x
            finish = y
        End Sub

        Public Overrides Function ToString() As String
            Return ("(" & start & "," & finish & ")")
        End Function

    End Class

    Class MyReturnStruct
        Public xValue As Integer
        Public yValue As Integer

        Public Sub New(ByVal x As Integer, ByVal y As Integer)
            xValue = x
            yValue = y
        End Sub

        Public Overrides Function ToString() As String
            Return ("(" & xValue & "," & yValue & ")")
        End Function

    End Class

    Delegate Function ThreadFunctionDelegate(ByVal arg1 As Integer, ByVal arg2 As
     MyStruct, ByVal arg3 As String) As MyReturnStruct

    Function ThreadFunction(ByVal arg1 As Integer, ByVal arg2 As MyStruct, ByVal
     arg3 As String) As MyReturnStruct
        System.Console.WriteLine("Thread {0} has arguments {1},{2}, {3}.",
                                Thread.CurrentThread.ManagedThreadId, arg1, arg2.
     ToString(), arg3)

        Return (New MyReturnStruct(arg2.start * 2, arg2.finish * 2))

    End Function

    Sub OnThreadFunctionComplete(ByVal result As IAsyncResult)
        Dim del As ThreadFunctionDelegate = result.AsyncState
        Dim val As MyReturnStruct = del.EndInvoke(result)

        Console.WriteLine("Thread {0} getting return value from ThreadFunction = 
    {1}.",
                      Thread.CurrentThread.ManagedThreadId, val.ToString())

    End Sub
    Sub Main()
        Dim d1 As ThreadFunctionDelegate = New ThreadFunctionDelegate(AddressOf 
    ThreadFunction)
        Dim d2 As ThreadFunctionDelegate = New ThreadFunctionDelegate(AddressOf 
    ThreadFunction)
        Dim d3 As ThreadFunctionDelegate = New ThreadFunctionDelegate(AddressOf 
    ThreadFunction)

        Console.WriteLine("Main Thread, Id = {0}.", Thread.CurrentThread.
    ManagedThreadId)

        d1.BeginInvoke(1, New MyStruct(50, 100), "String 1", New 
    AsyncCallback(AddressOf OnThreadFunctionComplete), d1)
        d2.BeginInvoke(2, New MyStruct(100, 150), "String 2", New 
    AsyncCallback(AddressOf OnThreadFunctionComplete), d2)
        d3.BeginInvoke(3, New MyStruct(150, 200), "String 3", New 
    AsyncCallback(AddressOf OnThreadFunctionComplete), d3)

        Thread.SpinWait(10000000)

        Console.ReadLine()

        Console.WriteLine("Done")

    End Sub

End Module
```

Figure 8-4 shows the output of the preceding example.



```
Main Thread, Id = 1.
Thread 3 has arguments 1,(50,100), String 1.
Thread 3 getting return value from ThreadFunction = (100,200).
Thread 3 has arguments 2,(100,150), String 2.
Thread 3 getting return value from ThreadFunction = (200,300).
Thread 3 has arguments 3,(150,200), String 3.
Thread 3 getting return value from ThreadFunction = (300,400).
```

Figure 8-4: Asynchronous delegates return data.

The preceding example now has a way to execute a function asynchronously on the thread pool and get data returned from this function. In my development, I use this pattern over and over because I expect my applications to do many things at once and still remain responsive to the user.

In Book IV, Chapter 9, I show you an application of the BeginInvoke, EndInvoke pattern that is used to send and receive data asynchronously across a network.

In Chapter 10 of Book IV, I show you enhancements to threading introduced in the .NET 4 framework called the Task Parallel Library (TPL). This library provides an even more convening model for executing methods asynchronously and even nesting asynchronous methods.

Keeping Things on Schedule with Timers

Almost any application I write has certain tasks that must be executed on a given schedule. For example, an application may need to check for updates everyday at midnight or automatically save a copy of a word-processing document every five minutes so that users don't lose hours of work if their computer crashes in the middle of writing a last-minute term paper. When I write applications, I don't want the main code to have to worry about maintaining the schedule. I just want to set a timer for a given interval and give it a task to do when the time expires. I also want this timer to operate in the background so that my threads that have to do work can do so without being blocked by the timer.

The Timer class in the System.Threading namespace makes it very easy to create timers and execute callback methods when a timer expires. The .NET framework simply executes your callback method on a thread in the thread pool when the timer expires. The framework takes care of all the details for you.

A simple example of an application that must execute tasks on a given schedule is a clock. A clock must update its digital display or, if it's an analog clock, must move the second, hour, and minute hands appropriately. What better example to illustrate timers in the .NET framework than with a simple digital clock that updates its display with the current time?

ON THE WEB

You can download the complete program (both C# and VB versions) for this example from this book's companion Web site at www.dummies.com/go/vs2010.

Creating a timer is simple in the .NET framework:

C#:

```
    _Timer = new System.Threading.Timer(new TimerCallback(OnTimerTick),n
ull,1000,1000);
```

VB:

```
_Timer = New System.Threading.Timer(New TimerCallback(AddressOf OnTimerTick),
Nothing, 1000, 1000)
```

The first argument is a TimerCallback delegate containing the function you want to execute when the timer expires. The second argument is for an object you want to pass to the TimerCallback function. The third argument is the time delay in milliseconds before the OnTimerTick is called, and the last argument is the timeout in milliseconds before each call to OnTimerTick. Because this is a digital clock that updates every second, setting this timeout to 1,000 milliseconds is sufficient. In the OnTimerTick, you get the current time and tell the application window to redraw itself with the current time:

C#:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Drawing2D;
using System.Threading;
using System.Drawing.Text;

namespace SimpleTimerCS
{
    public partial class Form1 : Form
    {
        private DateTime _CurrentTime = DateTime.Now;
        private System.Threading.Timer _Timer;
        private Bitmap _BackBuffer;
        public Form1()
        {
            InitializeComponent();

            _Timer = new System.Threading.Timer(new TimerCallback(OnTimerTick),
    null,1000,1000);
        }

        private void OnTimerTick(object o)
        {
            _CurrentTime = DateTime.Now;
            this.Invalidate();
        }

        protected override void OnPaintBackground(PaintEventArgs e)
        {
        }

        protected override void OnPaint(PaintEventArgs e)
        {
            if (_BackBuffer == null)
            {
                _BackBuffer = new Bitmap(e.ClipRectangle.Width, e.ClipRectangle.
    Height);
            }
            using (System.Drawing.Font textFont = new Font(FontFamily.
    GenericSansSerif, 36, FontStyle.Bold))
            {
```

```
            StringBuilder builder = new StringBuilder();

            Graphics g = Graphics.FromImage(_BackBuffer);

            g.SmoothingMode = SmoothingMode.HighQuality;
            g.TextRenderingHint = TextRenderingHint.AntiAliasGridFit;

            g.FillRectangle(Brushes.DodgerBlue, e.ClipRectangle);

            builder.AppendFormat("{0:HH:mm:ss}", _CurrentTime);

            StringFormat sf = new StringFormat();
            sf.Alignment = StringAlignment.Center;

            Rectangle centered = e.ClipRectangle;
            centered.Offset(0, (int)(e.ClipRectangle.Height - e.Graphics.
    MeasureString(builder.ToString(), textFont).Height) / 2);

            g.DrawString(builder.ToString(), textFont, Brushes.Yellow,
    centered, sf);

            g.Dispose();

            e.Graphics.DrawImageUnscaled(_BackBuffer, 0, 0);
          }
        }
    }
}
```

VB:

```
Imports System
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Drawing.Text
Imports System.Threading
Imports System.Text

Public Class Form1

    Private _Timer As System.Threading.Timer
    Private _CurrentTime As System.DateTime = System.DateTime.Now
    Private _BackBuffer As Bitmap

    Protected Overrides Sub OnPaintBackground(ByVal e As System.Windows.Forms.
    PaintEventArgs)

    End Sub

    Protected Overrides Sub OnPaint(ByVal e As System.Windows.Forms.
    PaintEventArgs)

        If _BackBuffer Is Nothing Then
            _BackBuffer = New Bitmap(e.ClipRectangle.Width, e.ClipRectangle.
    Height)
        End If


        Using textFont As New Font(FontFamily.GenericSansSerif, 36, FontStyle.
    Bold)
```

```
            Dim g As Graphics = Graphics.FromImage(_BackBuffer)

            g.SmoothingMode = SmoothingMode.HighQuality
            g.TextRenderingHint = TextRenderingHint.AntiAliasGridFit

            g.FillRectangle(Brushes.DodgerBlue, e.ClipRectangle)

            Dim builder As StringBuilder = New StringBuilder()

            builder.AppendFormat("{0:HH:mm:ss}", _CurrentTime)

            Dim sf As StringFormat = New StringFormat()

            sf.Alignment = StringAlignment.Center
            Dim centered As Rectangle = e.ClipRectangle

            centered.Offset(0, (e.ClipRectangle.Height - e.Graphics.
    MeasureString(builder.ToString(), textFont).Height) / 2)

            g.DrawString(builder.ToString(), textFont, Brushes.Yellow, centered,
    sf)

            g.Dispose()

            e.Graphics.DrawImageUnscaled(_BackBuffer, 0, 0)
        End Using

        MyBase.OnPaint(e)
    End Sub

    Private Sub OnTimerTick(ByVal o As Object)
        _CurrentTime = DateTime.Now
        Me.Invalidate()
    End Sub
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles MyBase.Load
        _Timer = New System.Threading.Timer(New TimerCallback(AddressOf
    OnTimerTick), Nothing, 1000, 1000)

    End Sub
End Class
```

Pressing Ctrl+F5 in Visual Studio launches the application, and the following window appears in Figure 8-5.

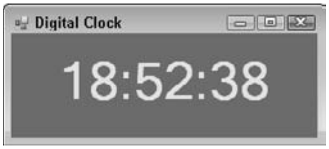


Figure 8-5: A simple digital clock.

Threads and timers are powerful tools in your programming toolkit for developing high-quality applications that can perform many tasks at once while providing a quality user experience for the users of your software.

Chapter 9: Writing Network Applications

In This Chapter

- Using sockets for network communications
- Creating an application to access content on the Web
- Creating a simple chat application

Since the mid-1990s, the Internet and the World Wide Web have become ubiquitous. The increased bandwidth capabilities allow for vast amounts of information to travel around the world in lightning-fast speeds. E-mail, pictures, documents, bank and market transactions, and even movies and television programs are transported from one place to another on the Internet. Now, more than ever, the world has become smaller and more connected.

Very few applications written today don't involve some sort of network connectivity. From applications that automatically download updates from the vendor's Web site or display the latest news headlines from organizations around the word, the ability to add networking capabilities to your applications is a must for nearly all software developers.

To help you build networked applications, the .NET framework offers libraries that help you network your applications with ease. The System.Net namespace contains all the classes you need to add any kind of networking to your application.

You can find the C# and VB examples for this chapter at `www.dummies.com/go/vs2010`.

Creating an Application to Access Content on the Web

Perhaps one of the most common types of network applications is one that accesses content on the Web using the Hypertext Transport Protocol (HTTP). HTTP is the means by which content from the Web is delivered to your browser. For an example of using HTTP to access content on the Web, you create a Windows Forms application that accesses weather forecast data for a particular city from Google using the Google Weather API. You use the Google Weather API to get the weather for Indianapolis, Indiana.

The Google Weather API provides forecast data in an XML file format. You access data for Indianapolis with the following URL: `www.google.com/ig/api?weather=Indianapolis`.

1. **From Visual Studio, choose File➪New➪Project.**

   The New Project dialog appears.

2. **Select Windows Forms Application from the C# program templates and click OK.**

   Visual Studio creates the Windows Forms application.

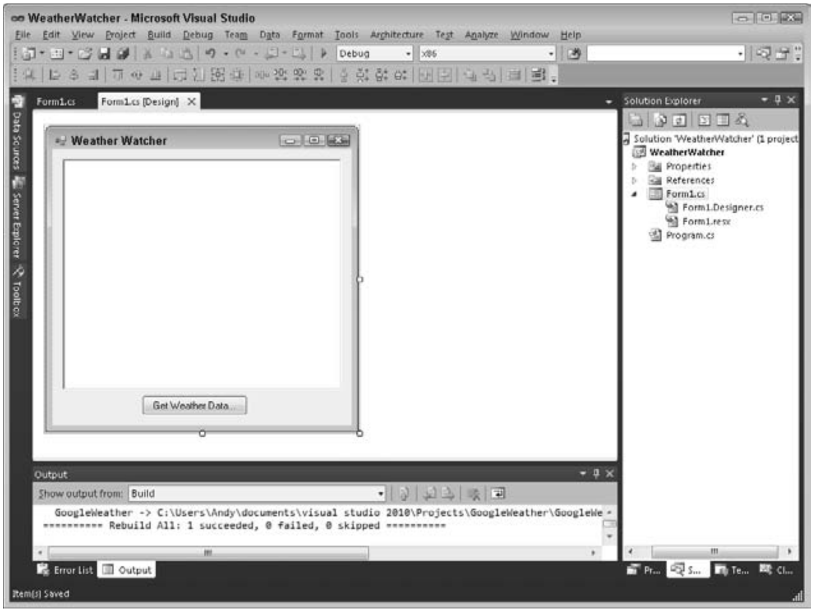
3. **Right click the References node in Solution Explorer.**

   The Add Reference dialog appears.

4. **Using the .NET tab on the Add Reference dialog box, choose `System.Net.dll` and click OK.**

   This step adds a reference to `System.Net.dll` in the References node of the Solution Explorer and allows the application to use the services provided by this assembly.

5. **In the Form Designer, drag a rich text box and a button from the Toolbox on your form, as shown in Figure 9-1.**



```
  GoogleWeather -> C:\Users\Andy\documents\visual studio 2010\Projects\GoogleWeather\GoogleWe
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

Figure 9-1: Form Designer for a sample weather application.

6. **Click the Get Weather Data button so that code for the Click Event is created.**

7. **To access weather data, add the code shown in Listing 9-1.**

Listing 9-1: C# and Visual Basic Code to Access Google Weather Data

C#:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Net;
using System.IO;
using System.Windows.Forms;

namespace WeatherWatcher
{
    public partial class Form1 : Form
    {
        static readonly string _WeatherUri =
           "http://www.google.com/ig/api?weather=Indianapolis";

        public Form1()
        {
            InitializeComponent();
        }

        private void _ButtonWeatherData_Click(object sender, EventArgs e)
        {
            try
            {
                HttpWebRequest request =
                    (HttpWebRequest)WebRequest.Create(_WeatherUri);

                HttpWebResponse response = (HttpWebResponse)request.
    GetResponse();

                Encoding utf8Encoding = Encoding.UTF8;

                StreamReader responseStream = new StreamReader(response.
    GetResponseStream(), utf8Encoding);
                string content = responseStream.ReadToEnd();

                StringBuilder builder = new StringBuilder();


                builder.AppendFormat("Google Weather for Indianapolis\n\
    n\n{0}.",
                                     new object[] { content });

                this.richTextBox1.Text = builder.ToString();

            }
            catch (System.Exception ex)
```

(continued)

Listing 9-1 ***(continued)***

```
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
}
```

VB:

```
Imports System.Net
Imports System.IO
Imports System
Imports System.Text


Public Class Form1
    Dim _WeatherUri As String = New String("http://www.google.com/ig/
    api?weather=Indianapolis")

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button1.Click
        Dim request As HttpWebRequest = WebRequest.Create(_WeatherUri)

        Dim response As HttpWebResponse = request.GetResponse()
        Dim utf8Encoding As Encoding = Encoding.UTF8

        Dim responseStream As StreamReader = New StreamReader(response.
    GetResponseStream(), utf8Encoding)
        Dim content As String = responseStream.ReadToEnd()

        Dim builder As StringBuilder = New StringBuilder()

        builder.AppendFormat("Google Weather for Indianapolis" & vbCrLf &
    vbCrLf & vbCrLf & "{0}.", content)

        Me.RichTextBox1.Text = builder.ToString()


    End Sub
End Class
```

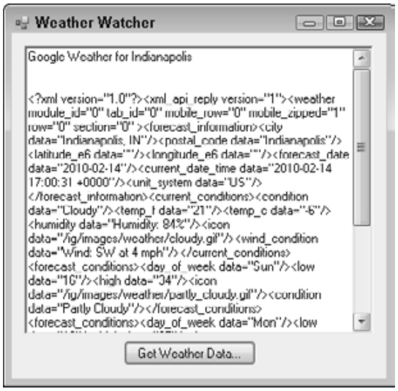
8. **Use Ctrl+F5 to run the application.**

9. **Click the Get Weather Data button.**

   The weather forecast data for Indianapolis appears in XML format, as shown in Figure 9-2.

Many things are happening in the preceding lines of code. First, the application has to make a TCP/IP connection to `www.google.com`.

The Transport Control Protocol/Internet Protocol (TCP/IP) is the backbone of the modern Internet. This protocol is the basis for the common application protocols used to transport data across the Internet such as Hypertext Transport Protocol (HTTP) and the File Transport Protocol (FTP). You can find out more about network protocols on Wikipedia at `http://en.wikipedia.org/wiki/Internet_protocol_suite`.



```
Google Weather for Indianapolis


<?xml version="1.0"?><xml_api_reply version="1"><weather
module_id="0" tab_id="0" mobile_row="0" mobile_zipped="1"
row="0" section="0" ><forecast_information><city
data="Indianapolis, IN"/><postal_code data="Indianapolis"/>
<latitude_e6 data=""/><longitude_e6 data=""/><forecast_date
data="2010-02-14"/><current_date_time data="2010-02-14
17:00:31 +0000"/><unit_system data="US"/>
</forecast_information><current_conditions><condition
data="Cloudy"/><temp_f data="21"/><temp_c data="-6"/>
<humidity data="Humidity: 84%"/><icon
data="/ig/images/weather/cloudy.gif"/> <wind_condition
data="Wind: SW at 4 mph"/> </current_conditions>
<forecast_conditions><day_of_week data="Sun"/><low
data="16"/><high data="34"/><icon
data="/ig/images/weather/partly_cloudy.gif"/><condition
data="Partly Cloudy"/></forecast_conditions>
<forecast_conditions><day_of_week data="Mon"/><low
```

Figure 9-2: Weather data from Google.

In the .NET framework, the `System.Net.Sockets.Socket` class creates a TCP/IP connection to a given location. In this example, the `System.Net.HttpWebRequest` class makes the socket connections for you. Once connected, the `System.Net.HttpWebRequest` issues a command to get data from the URL that you provided. After this request reaches the destination at `www.google.com`, this site creates a response and sends it back on the same TCP/IP connection. The `System.Net.HttpWebResponse` class handles the response from `www.google.com`. The `GetResponse` method waits for data from `www.google.com`, and when data are available, this method returns. Now you can read the data into a string and display in the `RichTextBox` on your form. The `HttpWebRequest` and `HttpWebResponse` classes handle all the details of making TCP/IP connections and handling HTTP requests and responses freeing the developer from concern about the details of the HTTP protocol.

You can see from the preceding example that the .NET framework makes it easy to add the capability to access data from the Web in your applications.

The preceding example, however, is a very naïve implementation. The main flaw in this implementation is that it calls `GetResponse` on the `HttpWebResponse` class, which blocks waiting for data to return from `www.google.com`. When accessing data across the network, many factors can impact the time required for data to return. The data provider may be in a distant location on a network with high latency, or the provider may have to execute lengthy operations before returning data. In any case, these factors are beyond the control of the requesting application. Because the request and response occur in the same thread as the `Click` event on the button, any updating of the application window is blocked until data arrive, and `GetResponse` returns. Blocking the window from updating isn't a good user experience; the application appears frozen because the user isn't able to drag the application on the desktop or maximize/minimize it.

Fortunately, the HttpWebResponse class provides asynchronous methods to get data from a data source. In this case, the request goes to www.google.com, but the application doesn't block waiting for data, but instead, the HttpWebResponse issues a callback when data arrives. At this point, you can read the data and assign the forecast data to the RichTextBox.

You can change your code in the Click event of the Get Weather Data button, as shown in Listing 9-2.

Listing 9-2: C# and Visual Basic with Asynchronous Request of Google Weather Data

C#:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Net;
using System.IO;
using System.Windows.Forms;

namespace WeatherWatcher
{
    public partial class Form1 : Form
    {
        static readonly string _WeatherUri =
           "http://www.google.com/ig/api?weather=Indianapolis";

        delegate void UpdateText(string s);

        public Form1()
        {
            InitializeComponent();
        }


        private void OnResponseReady(IAsyncResult result)
        {
            HttpWebRequest request = (HttpWebRequest)result.AsyncState;

            HttpWebResponse response = (HttpWebResponse) request.
    EndGetResponse(result);


            Encoding utf8Encoding = Encoding.UTF8;

            StreamReader responseStream = new StreamReader(response.
    GetResponseStream(), utf8Encoding);
            string content = responseStream.ReadToEnd();

            StringBuilder builder = new StringBuilder();
            builder.AppendFormat("Google Weather for Indianapolis\n\n\n{0}.",
                                 new object[] { content });

            UpdateText updateText = delegate(string s) { this.richTextBox1.Text =
    s; };

            this.richTextBox1.BeginInvoke(updateText,
                                          new object[] {builder.ToString()}
                                         );
        }

        private void _ButtonWeatherData_Click(object sender, EventArgs e)
        {
            try
            {
                HttpWebRequest request =
                    (HttpWebRequest)WebRequest.Create(_WeatherUri);

                IAsyncResult result = request.BeginGetResponse(new AsyncCallback(
    OnResponseReady),request);

            }
            catch (System.Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
}
```

VB:

```
Imports System
Imports System.IO
Imports System.Net
Imports System.Text

Public Class Form1
    Dim _WeatherUri As String = New String("http://www.google.com/ig/
    api?weather=Indianapolis")

    Private Sub OnResponseReady(ByVal result As IAsyncResult)

        Dim utf8Encoding As Encoding = Encoding.UTF8
        Dim request As HttpWebRequest = result.AsyncState
        Dim response As HttpWebResponse = request.EndGetResponse(result)

        Dim responseStream As StreamReader = New StreamReader(response.
    GetResponseStream(), utf8Encoding)
        Dim content As String = responseStream.ReadToEnd()

        Dim builder As StringBuilder = New StringBuilder()

        builder.AppendFormat("Google Weather for Indianapolis" & vbCrLf & vbCrLf
    & vbCrLf & "{0}.", content)

        Me.RichTextBox1.Text = builder.ToString()

    End Sub
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles Button1.Click
```

(continued)

Listing 9-2 *(continued)*

```
        Dim request As HttpWebRequest = WebRequest.Create(_WeatherUri)

        Dim asyncResult As IAsyncResult = request.BeginGetResponse(New System.
    AsyncCallback(AddressOf OnResponseReady), request)

    End Sub
End Class
```

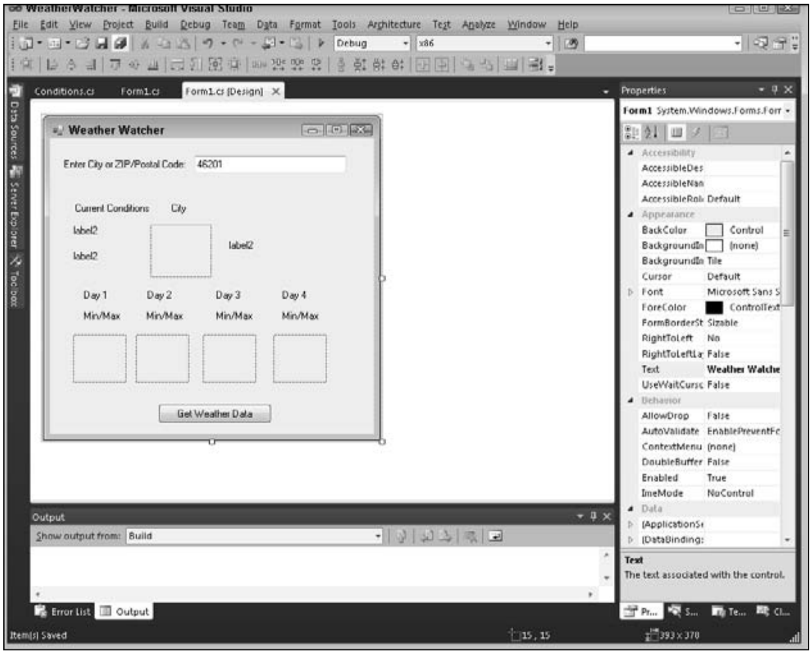
This time, when you click the Get Weather Data button, the code makes a call to `BeginGetResponse` and provides an `AsyncCallback` parameter. This parameter is a delegate that the `HttpWebResponse` class calls when data arrive from `www.google.com`. In this example, the `HttpWebResponse` class calls `OnResponseReady` when data are available. Under the covers, the .NET framework is executing the request on a thread pool thread and returning the result on this same thread, freeing the main application thread so that the window can redraw, maximize, or minimize, which provides a satisfying user experience. Book IV, Chapter 8 provides more details about threads and thread pools.

The second example is much better. You can move the window around while the request is processed, but displaying the raw XML weather data is of limited utility. Deciphering the forecast from the XML data visually isn't intuitive. Also, the request URL is hard-coded to return weather data for Indianapolis, Indiana only. If you're planning a trip to another city, you'd want to see the weather forecast for that city to know what clothing to pack and whether your golf plans are in jeopardy of being rained out.

A better example would provide a graphical representation of the weather data in the application window. Fortunately, the .NET framework makes this task rather easy. Figure 9-3 shows an example Form Designer layout for a simple weather forecast application. This form contains a `PictureBox` for displaying the appropriate icon for the current conditions as well as a four-day forecast. Also, labels display the current temperature, humidity, forecast time, and min/max temperatures for the next four days.

Now, all you have to do is parse the XML data from Google and populate the labels and `PictureBox` objects with the appropriate data. Press Ctrl+F5 to run the application, and you see weather data similar to Figure 9-4.

This application is much more useful because it now graphically displays the current conditions and four-day forecast for a given location. If you're planning a trip to visit friends in the Waltham, MA 02453 area, you can just enter the ZIP/Postal Code 02453 and click the Get Weather Data button to see the forecast for that location. Alternatively, you can just type the City in the text box and see the forecast for that city.



Enter City or ZIP/Postal Code: 46201

Current Conditions City

label2

label2

label2

Day 1 Day 2 Day 3 Day 4

Min/Max Min/Max Min/Max Min/Max

Show output from: Build

Properties

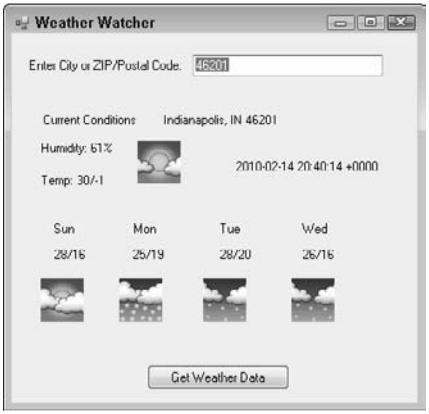
Form1 System.Windows.Forms.Forr

Accessibility	
AccessibleDes	
AccessibleNan	
AccessibleRol	Default
Appearance	
BackColor	Control
BackgroundIn	(none)
BackgroundIn	Tile
Cursor	Default
Font	Microsoft Sans S
ForeColor	ControlText
FormBorderSt	Sizable
RightToLeft	No
RightToLeftLa	False
Text	**Weather Watche**
UseWaitCursc	False
Behavior	
AllowDrop	False
AutoValidate	EnablePreventFc
ContextMenu	(none)
DoubleBuffer	False
Enabled	True
ImeMode	NoControl
Data	
(ApplicationSe	
(DataBinding:	

Text
The text associated with the control.

Item(s) Saved 15 , 15 393 x 370

Figure 9-3: Form Designer layout for weather watcher application.



Enter City or ZIP/Postal Code: 46201

Current Conditions Indianapolis, IN 46201

Humidity: 61%

Temp: 30/-1

2010-02-14 20:40:14 +0000

Sun	Mon	Tue	Wed
28/16	25/19	28/20	26/16

Figure 9-4: Weather watcher displays weather data.

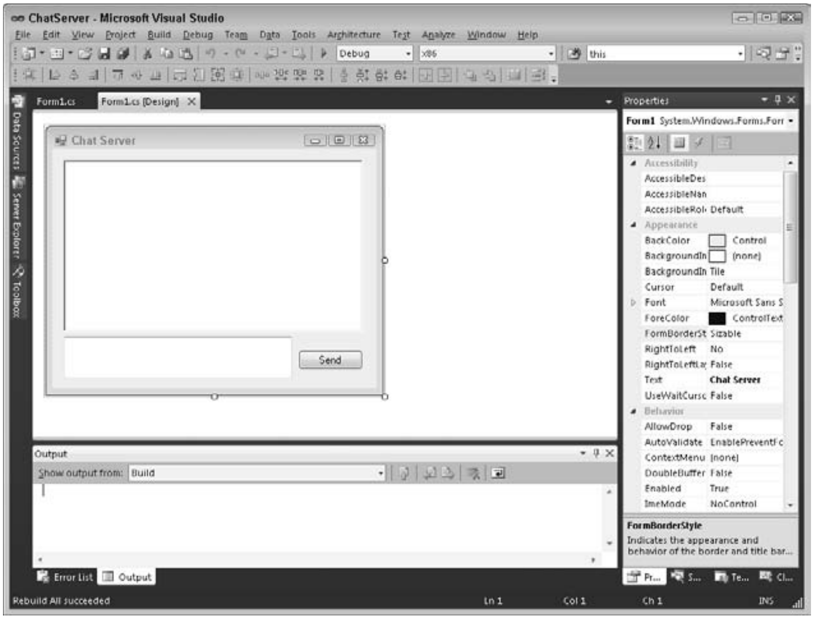
Creating a Simple Chat Application

One common network scenario in networked application is a client/server application. In this scenario, one application is the server and listens from incoming client application. The server responds to requests from the client, and the client receives responses from the server. A simple example of this type of application is a chat application.

In the next example, a Chat Server application listens for incoming connections, and a Chat Client application connects to Chat Server. Once connected, both applications can exchange text messages.

The Chat Server Application

The Chat Server application is a simple Windows Forms application. Using the Form Designer, create a window like shown in Figure 9-5. See Book IV, Chapter 2 for examples creating Windows Forms applications.



Form1 System.Windows.Forms.Forr

Accessibility	
AccessibleDes	
AccessibleNan	
AccessibleRol	Default
Appearance	
BackColor	Control
BackgroundIn	(none)
BackgroundIn	Tile
Cursor	Default
Font	Microsoft Sans S
ForeColor	ControlText
FormBorderSt	Sizable
RightToLeft	No
RightToLeftLa	False
Text	**Chat Server**
UseWaitCursc	False
Behavior	
AllowDrop	False
AutoValidate	EnablePreventFc
ContextMenu	(none)
DoubleBuffer	False
Enabled	True
ImeMode	NoControl

FormBorderStyle
Indicates the appearance and behavior of the border and title bar...

Figure 9-5: Chat Server Form Designer example.

This window contains a `RichTextBox` that contains the chat text, a small `TextBox` near the bottom of the window that will contain the text to send to the client, and a Send button to send the text to the client.

You create a server application by creating a socket using the `System.Net.Sockets.Socket` class and assigning an `IPEndPoint` object to this socket. The `IPEndPoint` object indicates the address and port number on which the server will listen for incoming connections. Once created, the `Listen` method on the `Socket` class is called to indicate that the server is listening for incoming connections. Listing 9-3 shows the code for establishing a server socket.

Listing 9-3: C# and Visual Basic Code to Create a Server That Listens for Connection

C#:

```
  private void Form1_Load(object sender, EventArgs e)
        {
            _Server = new Socket(AddressFamily.InterNetwork,
                                 SocketType.Stream,
                                 ProtocolType.Tcp);

            _Endpoint = new IPEndPoint(IPAddress.Any, 10255);

            _Server.Bind(_Endpoint);
            _Server.Listen(5);

            _Server.BeginAccept(new AsyncCallback(OnAcceptClient), _Server);
        }
```

VB:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles MyBase.Load
        _Server = New Socket(AddressFamily.InterNetwork,
                             SocketType.Stream,
                             ProtocolType.Tcp)

        _Endpoint = New IPEndPoint(IPAddress.Any, 10255)

        _Server.Bind(_Endpoint)
        _Server.Listen(5)

        _Server.BeginAccept(New AsyncCallback(AddressOf OnAcceptClient), _Server)
    End Sub
```

The BeginAccept method tells the server socket to start an asynchronous wait to accept incoming connections. When a client connects, the Socket class calls the OnAcceptClient method to complete the client connection, as shown in Listing 9-4.

Listing 9-4: C# and Visual Basic Code to Accept an Incoming Connection

C#:

```
private void OnAcceptClient(IAsyncResult result)
        {
            Socket server = (Socket)result.AsyncState;
            Socket client = server.EndAccept(result);

            _Client = client;

            StringBuilder builder = new StringBuilder();
```

(continued)

Listing 9-4 *(continued)*

```
        builder.AppendFormat("Accepted connection from client at - {0}" +
Environment.NewLine,
                             client.RemoteEndPoint.ToString());

        System.Diagnostics.Debug.WriteLine(builder.ToString());

        this.Invoke((Action)(() => this.richTextBox1.AppendText(builder.
ToString())));

        string welcomeData = "Welcome, you are connected to " + client.
RemoteEndPoint.ToString() + Environment.NewLine;

        byte[] message = Encoding.UTF8.GetBytes(welcomeData);

        client.BeginSend(message, 0, message.Length, SocketFlags.None,
                          new AsyncCallback(OnSendComplete), client);

        server.BeginAccept(new AsyncCallback(OnAcceptClient), server);

    }
```

VB:

```
Private Sub OnAcceptClient(ByVal result As IAsyncResult)
        Dim server As Socket = result.AsyncState
        Dim client = server.EndAccept(result)

        _Client = client

        Dim builder As StringBuilder = New StringBuilder()
        builder.AppendFormat("Accepted connection from client at - {0}" +
    Environment.NewLine,
                             client.RemoteEndPoint.ToString())

        System.Diagnostics.Debug.WriteLine(builder.ToString())

        Me.BeginInvoke(New WriteTextDelegate(AddressOf WriteText), builder.
    ToString())

        Dim welcomeData As String = "Welcome, you are connected to " + client.
    RemoteEndPoint.ToString() + Environment.NewLine
        Dim message As Byte() = Encoding.UTF8.GetBytes(welcomeData)

        client.BeginSend(message, 0, message.Length, SocketFlags.None,
                         New System.AsyncCallback(AddressOf OnSendComplete),
    client)


    End Sub
```

After the connection is complete, the server calls `BeginAccept` once again to wait for another client connection. After the connection occurs, data can flow back and forth between the client and the server. Listing 9-5 shows the server code to send and receive data from the client.

Listing 9-5: C# and Visual Basic Code to Exchange Data with the Client

C#:

```
private Socket _Server;
private Socket _Client;

private IPEndPoint _Endpoint;
private byte[] _Buffer = new byte[1024];
private int _Size = 1024;

private void OnReceiveData(IAsyncResult result)
        {
            Socket client = (Socket)result.AsyncState;

            int nBytes = client.EndReceive(result);

            if (nBytes == 0)
            {
                this.Invoke((Action)(() => this.richTextBox1.AppendText("Client
    at " + client.RemoteEndPoint.ToString() + " disconnected." + Environment.
    NewLine)));

                client.Close();

                return;

            }

            string data = Encoding.UTF8.GetString(_Buffer, 0, nBytes);

            this.Invoke((Action)(() => this.richTextBox1.AppendText("Client
    at " + client.RemoteEndPoint.ToString() + Environment.NewLine + data +
    Environment.NewLine)));

            client.BeginReceive(_Buffer, 0, _Size, SocketFlags.None,
                       new AsyncCallback(OnReceiveData), client);

        }

        private void OnSendComplete(IAsyncResult result)
        {
            Socket client = (Socket) result.AsyncState;

            System.Diagnostics.Debug.WriteLine("Sent data to client at " +
    client.RemoteEndPoint.ToString());

            client.BeginReceive(_Buffer, 0, _Size, SocketFlags.None,
                            new AsyncCallback(OnReceiveData), client);

        }
```

VB:

```
Private Sub OnSendComplete(ByVal result As IAsyncResult)
        Dim client As Socket = result.AsyncState

        System.Diagnostics.Debug.WriteLine("Sent data to client at " + client.
    RemoteEndPoint.ToString())
```

(continued)

Listing 9-5 *(continued)*

```
        client.BeginReceive(_Buffer, 0, _Size, SocketFlags.None,
                            New AsyncCallback(AddressOf OnReceiveData), client)
    End Sub

    Private Sub OnReceiveData(ByVal result As IAsyncResult)
        Dim client As Socket = result.AsyncState

        Dim nBytes As Integer = client.EndReceive(result)

        If (nBytes = 0) Then
            Me.BeginInvoke(New WriteTextDelegate(AddressOf WriteText), "Client
    at " + client.RemoteEndPoint.ToString() + " disconnected." + Environment.
    NewLine)

             client.Close()
            Return

        End If

        Dim data As String = Encoding.UTF8.GetString(_Buffer)

        Me.BeginInvoke(New WriteTextDelegate(AddressOf WriteText), "Client at " +
    client.RemoteEndPoint.ToString() + Environment.NewLine + data + Environment.
    NewLine)

        client.BeginReceive(_Buffer, 0, _Size, SocketFlags.None,
                      New AsyncCallback(AddressOf OnReceiveData), client)
    End Sub
```

Now, press Ctrl+F5 to launch the ChatServer application.

The Chat Client application

Now that the Chat Server is completed, you can develop the Chat Client. The Chat Client looks very similar to the Chat Server application. It also has a `RichTextBox` for displaying chat text along with a `TextBox` for entering data and a Send button.

The Chat Client code is simpler than the Chat Server. The Chat Client just needs to connect to the Chat Server, and then data can start flowing back and forth. The same `System.Net.Sockets.Socket` class used for the server socket is the same class used for the client socket. Listing 9-6 shows the code required to connect to Chat Server.

Listing 9-6: C# and Visual Basic Code for the Client to Connect to the Server

C#:

```
Socket _Socket;
byte[] _Buffer = new byte[1024];
int _BufferSize = 1024;
```

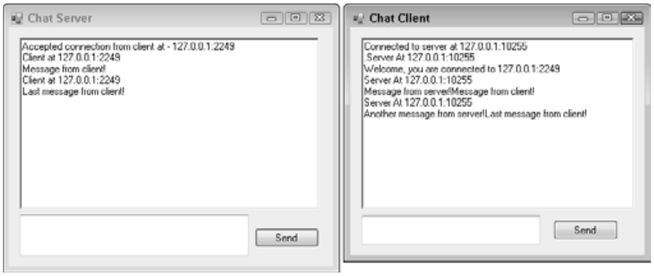
```
            _Socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
     ProtocolType.Tcp);

             EndPoint endPoint = new IPEndPoint(IPAddress.Loopback, 10255);

             _Socket.BeginConnect(endPoint, new AsyncCallback(OnConnectComplete),
     _Socket);

private void OnConnectComplete(IAsyncResult result)
        {
            Socket client = (Socket)result.AsyncState;

            client.EndConnect(result);

            StringBuilder builder = new StringBuilder();

            builder.AppendFormat("Connected to server at {0}\n.",
                                 client.RemoteEndPoint.ToString());

            this.BeginInvoke((Action)(() => this.richTextBox1.AppendText(builder.
     ToString())));
            this.BeginInvoke((Action)(() => this.button1.Enabled = true));

            client.BeginReceive(_Buffer, 0, _BufferSize, SocketFlags.None,
                                new AsyncCallback(OnReceiveComplete), client);

        }
```

VB:

```
Private _Socket As Socket
    Dim _Buffer As Byte() = New Byte(1024) {}
    Dim _BufferSize As Integer = 1024

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.
     EventArgs) Handles MyBase.Load
        _Socket = New Socket(AddressFamily.InterNetwork, SocketType.Stream,
     ProtocolType.Tcp)

        Dim endPoint As EndPoint = New IPEndPoint(IPAddress.Loopback, 10255)

        _Socket.BeginConnect(endPoint, New AsyncCallback(AddressOf
     OnConnectComplete), _Socket)
    End Sub
Private Sub OnConnectComplete(ByVal result As IAsyncResult)
        Dim client As Socket = result.AsyncState

        client.EndConnect(result)

        Dim builder As StringBuilder = New StringBuilder()

        builder.AppendFormat("Connected to server at {0}" + Environment.NewLine,
                                client.RemoteEndPoint.ToString())
        'Me.button1.Enabled = True

        Me.BeginInvoke(New WriteTextDelegate(AddressOf WriteText), builder.
     ToString())

        client.BeginReceive(_Buffer, 0, _BufferSize, SocketFlags.None,
                           New AsyncCallback(AddressOf OnReceiveComplete),
     client)

    End Sub
```

After the connection occurs, the code to send and receive data is identical to that in the Chat Server application.

Now, press Ctrl+F5 to run the Chat Client application. The Chat Server and Chat Client are running and ready to exchange data. Figure 9-6 shows example data exchanged between Chat Server and Chat Client.



```
Accepted connection from client at - 127.0.0.1:2249
Client at 127.0.0.1:2249
Message from client!
Client at 127.0.0.1:2249
Last message from client!
```

```
Connected to server at 127.0.0.1:10255
 Server At 127.0.0.1:10255
Welcome, you are connected to 127.0.0.1:2249
Server At 127.0.0.1:10255
Message from server!Message from client!
Server At 127.0.0.1:10255
Another message from server!Last message from client!
```

Figure 9-6: Chat Server and Chat Client applications.

The .NET framework makes networking your applications very simple. After you have a few networking concepts under your belt, you can harness the .NET framework to allow your applications to get data from the Web or to pass information from one application to the next creating the foundation for a larger-scale distributed system.

Chapter 10: Parallel Programming

In This Chapter

- **Discovering the .NET Task Parallel Library (TPL)**
- **Executing tasks in parallel in unmanaged C++ with the Parallel Patterns Library (PPL)**
- **Dividing up the work with Parallel Language Integrated Query (PLINQ)**

Most personal and notebook computers today come equipped with multiple processors or multicore processors that can execute instructions in parallel on each processor or processor core. In fact, personal computers containing processors with four cores are commonplace today, allowing the computer to execute four sets of instructions simultaneously. Finally, with the release of Visual Studio 2010, programming libraries have become available that allow you to break down a process or algorithm into separate tasks that can be executed on different processors or processor cores.

In this chapter, you explore two parallel programming libraries that are new in .NET 4. The first is the Task Parallel Library (TPL), which allows you to perform the same operation in parallel on items in a collection or array as well as execute multiple tasks in parallel. You also become familiar with the Parallel Language Integrated Query (PLINQ), which allows you to execute queries in parallel on a given data collection. For the developer of unmanaged C++ applications, a new Concurrency Runtime library provides a Parallel Patterns Library (PPL) that is analogous to the TPL in .NET.

Introducing the .NET Task Parallel Library

This separation of tasks onto different processors is what is known as *parallel programming.* The .NET Task Parallel Library allows you to perform two kinds of parallelism in your application:

- *Data parallelism* is the concept of performing the same operation on a given set of data in parallel — for example, traversing a list of 1,000 files in a directory and copying them to another directory.
- *Task parallelism* is the concept of executing many different tasks simultaneously.

The TPL provides for both models of parallelism. The .NET framework provides the TPL in the `System.Threading.Tasks` namespace that is part of the `System.Core.dll` assembly.

Examining data parallelism in the TPL

In introductory programming classes, instructors often assign exercises to students that teach them to write code that performs operations over a given set of data and collect results of these operations and displays them to the screen. One such example is the Towers of Hanoi mathematical game consisting of three rods and several different-sized disks that can slide onto any rod.

The game starts with the disks in a stack on one of the rods, with the stack based on ascending order of size. The objective is to move the entire stack to another rod by moving only one disk at a time, ensuring that no disk may be placed on a smaller disk.

The following equation gives the minimum number of steps required to complete the game with *n* disks.

$$\boldsymbol{h}anoi(n)=\begin{cases} \square \\ 1 \; if \;\; n=\mathbf{1} \\ 2\cdot \boldsymbol{h}anoi(n-\mathbf{1})+1 \; if \;\; n>\mathbf{1} \end{cases}$$

You can download the complete program (both C# and VB versions) for this example from this book's companion Web site at `www.dummies.com/go/vs2010`.

The following listings show an implementation for this algorithm in C# and Visual Basic computing the number of steps given 4, 22, 19, 24, 6, 11, 29, and 14 disks.

C#:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace TowerOfHanoiCS
{
    class Program
    {
        static int hanoi(int n)
        {
            if (n == 1)
            {
                return (1);
            }
            else
            {
                return (2 * hanoi(n - 1) + 1);
            }
        }
        static void Main(string[] args)
        {
            List<int> inputs = new List<int>();

            Dictionary<int, int> serialResults = new Dictionary<int,
    int>(EqualityComparer<int>.Default);

            inputs.Add(4);
            inputs.Add(22);
            inputs.Add(19);
            inputs.Add(24);
            inputs.Add(6);
            inputs.Add(11);
            inputs.Add(29);
            inputs.Add(14);

            /*Serial Version*/
            inputs.ForEach(n =>
                           {
                               serialResults.Add(n, hanoi(n));
                           }
                          );


            foreach (KeyValuePair<int, int> pair in serialResults)
            {
                Console.WriteLine("hanoi({0}) = {1}", pair.Key, pair.Value);
            }

            Console.WriteLine();


        }
    }
}
```

VB:

```
Imports System
Imports System.Collections.Generic
Imports System.Text
Imports System.Threading
Imports System.Threading.Tasks

Module Module1

    Function hanoi(ByVal n As Integer)
        If (n = 1) Then
            Return (1)
        End If

        Return (2 * hanoi(n - 1) + 1)

    End Function

    Sub Main()
        Dim inputs As List(Of Integer) = New List(Of Integer)

        Dim serialResults As Dictionary(Of Integer, Integer) = New Dictionary(Of
    Integer, Integer)()
```

```
        inputs.Add(4)
        inputs.Add(22)
        inputs.Add(19)
        inputs.Add(24)
        inputs.Add(6)
        inputs.Add(11)
        inputs.Add(29)
        inputs.Add(14)

        'Serial Version
        inputs.ForEach(Sub(n As Integer)
                           serialResults.Add(n, hanoi(n))
                       End Sub
                      )



        For Each pair As KeyValuePair(Of Integer, Integer) In serialResults
            Console.WriteLine("hanoi({0}) = {1}", pair.Key, pair.Value)
        Next pair

        Console.WriteLine()


    End Sub

End Module
```

In the preceding example, the calculation minimum number of steps to complete the game for each value for the number of disks executes sequentially on the same thread that executes the `ForEach` method on the list of inputs. For recursive algorithms, such as the Towers of Hanoi, the computation can be time-consuming, so performing these tasks in parallel would be nice. In the Towers of Hanoi algorithm, the computation of minimum number of steps given four disks and the computation given 22 disks do not depend on one another. The TPL can help out by separating each computation for this algorithm onto a separate processing unit. The TPL in `System.Threading.Tasks` provides a `Parallel.For` and `Parallel.ForEach` method that allows each iteration of the loop over the inputs to the algorithm to execute in parallel. To make the preceding algorithm execute in parallel, change the `ForEach` loop to a `Parallel.ForEach` loop:

C#:

```
Dictionary<int, int> parallelResults = new Dictionary<int,
    int>(EqualityComparer<int>.Default);

Parallel.ForEach(inputs, x =>
                             {
                                 parallelResults.Add(x, hanoi(x));
                             }
                            );
```

VB:

```
Dim parallelResults As Dictionary(Of Integer, Integer) = New Dictionary(Of
    Integer, Integer)()
```

```
Parallel.ForEach(inputs, Sub(n As Integer)
                                     parallelResults.Add(n, hanoi(n))
                                 End Sub
                        )
```

The complete listings shows both serial and parallel versions of the computations:

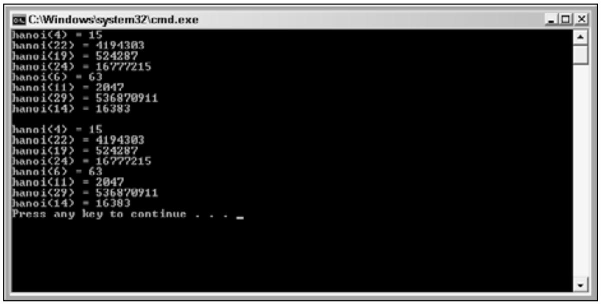
C#:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace TowerOfHanoiCS
{
    class Program
    {
        static int hanoi(int n)
        {
            if (n == 1)
            {
                return (1);
            }
            else
            {
                return (2 * hanoi(n - 1) + 1);
            }
        }

        static void Main(string[] args)
        {
            List<int> inputs = new List<int>();

            Dictionary<int, int> serialResults = new Dictionary<int,
    int>(EqualityComparer<int>.Default);
            Dictionary<int, int> parallelResults = new Dictionary<int,
    int>(EqualityComparer<int>.Default);

            inputs.Add(4);
            inputs.Add(22);
            inputs.Add(19);
            inputs.Add(24);
            inputs.Add(6);
            inputs.Add(11);
            inputs.Add(29);
            inputs.Add(14);

            /*Serial Version*/
            inputs.ForEach(n =>
                           {
                               serialResults.Add(n, hanoi(n));
                           }
                          );

            /*Parallel Version*/ Parallel.ForEach(inputs, x =>
                             {
                                 parallelResults.Add(x, hanoi(x));
                             }
                            );



            foreach (KeyValuePair<int, int> pair in serialResults)
            {
                Console.WriteLine("hanoi({0}) = {1}", pair.Key, pair.Value);
            }

            Console.WriteLine();

            foreach (KeyValuePair<int, int> pair in parallelResults)
            {
                Console.WriteLine("hanoi({0}) = {1}", pair.Key, pair.Value);
            }

        }
    }
}
```

VB:

```
Imports System
Imports System.Collections.Generic
Imports System.Text
Imports System.Threading
Imports System.Threading.Tasks

Module Module1

    Function hanoi(ByVal n As Integer)
        If (n = 1) Then
            Return (1)
        End If

        Return (2 * hanoi(n - 1) + 1)

    End Function

    Sub Main()
        Dim inputs As List(Of Integer) = New List(Of Integer)

        Dim serialResults As Dictionary(Of Integer, Integer) = New Dictionary(Of
    Integer, Integer)()
        Dim parallelResults As Dictionary(Of Integer, Integer) = New
    Dictionary(Of Integer, Integer)()

        inputs.Add(4)
        inputs.Add(22)
        inputs.Add(19)
        inputs.Add(24)
        inputs.Add(6)
        inputs.Add(11)
        inputs.Add(29)
        inputs.Add(14)

        'Serial Version
        inputs.ForEach(Sub(n As Integer)
                           serialResults.Add(n, hanoi(n))
                       End Sub
                      )

        'Parallel Version
        Parallel.ForEach(inputs, Sub(n As Integer)
                                     parallelResults.Add(n, hanoi(n))
                                 End Sub
                        )

        For Each pair As KeyValuePair(Of Integer, Integer) In serialResults
            Console.WriteLine("hanoi({0}) = {1}", pair.Key, pair.Value)
        Next pair

        Console.WriteLine()

        For Each pair As KeyValuePair(Of Integer, Integer) In parallelResults
            Console.WriteLine("hanoi({0}) = {1}", pair.Key, pair.Value)
        Next pair

    End Sub

End Module
```

Figure 10-1 shows the output of the preceding example.



```
hanoi(4) = 15
hanoi(22) = 4194303
hanoi(19) = 524287
hanoi(24) = 16777215
hanoi(6) = 63
hanoi(11) = 2047
hanoi(29) = 536870911
hanoi(14) = 16383

hanoi(4) = 15
hanoi(22) = 4194303
hanoi(19) = 524287
hanoi(24) = 16777215
hanoi(6) = 63
hanoi(11) = 2047
hanoi(29) = 536870911
hanoi(14) = 16383
Press any key to continue . . . _
```

Figure 10-1: Towers of Hanoi sample application.

Examining task parallelism in the TPL

ON THE WEB

You can download the complete program (both C# and VB versions) for this example from this book's companion Web site at www.dummies.com/go/vs2010.

Task parallelism is the concept of executing one or more tasks asynchronously on separate processors or processor cores. In some cases, you may want the tasks to execute and not wait for the task to complete. In other cases, you may want the function that executes the task to wait until the task completes and then continue execution. Or you may want the function that executes the task to go off and do something else while the task executes and notify you when the task completes. The TPL provides convenient models for each of these situations.

Issuing requests without waiting for a response

A common scenario in modern software development is creating a task that goes out to the Web and downloads data and returns that data to your application for further processing. Simply making a request to the Web site and waiting for it to respond and return the data isn't an ideal solution.

For example, if you have a simple Windows Forms application that makes a request to a Web site when the user presses a button, if the function that executes in response to the button press makes a request to the Web site and waits for a response that potentially may take several seconds or longer, the button remains pressed, and the application appears frozen. This situation doesn't make many users happy. Ideally, you want the code that handles the button press to issue the request to the Web site and then allow the calling function to go off and do other things, such as respond when you drag the application window around.

In the System.Net namespace, the WebRequest class that allows you to issue requests to Web sites provided asynchronous methods, such as BeginGetResponse, which didn't block the caller waiting for a response, but instead called a provided callback when the Web site returned the requested data. These methods provided the needed asynchronous model for performing potentially long-running operations, but were cumbersome to use since the programmer was required to call methods such as EndGetResponse to get the data returned by the request. The TPL simplifies this model with the Task class in the System.Threading.Tasks namespace. The Task class is a generic class whose generic parameter is a data type that is returned by an asynchronous operation.

This example illustrates the use of the asynchronous methods of the HttpWebRequest class to get weather data from Google. The TPL allows you to encapsulate the BeginGetResponse and EndGetResponse methods into a single Task object as shown in the following listing:

C#:

```
HttpWebRequest request =
               (HttpWebRequest)WebRequest.Create(url);

Task<WebResponse> task =
               Task<WebResponse>.Factory.FromAsync(request.BeginGetResponse,
                                       request.EndGetResponse,
                                       null);
```

VB:

```
Dim request As HttpWebRequest = WebRequest.Create(url)
        Dim myTask As Task(Of WebResponse) = Task(Of WebResponse).Factory.
    FromAsync(AddressOf request.BeginGetResponse,

    AddressOf request.EndGetResponse,

    Nothing)
```

Now, the BeginGetResponse and EndGetResponse are packaged into a single Task object that you can use to control the execution of the request to the Web site. Some common methods on the task class appear in Table 10-1.

Table 10-1 Common Methods for System.Threading. Tasks.Task Class

Method	*Description*
ContinueWith	Creates a continuation that executes when the task completes
RunSynchronously	Runs the task synchronously on the current TaskScheduler
RunSynchronously (TaskScheduler)	Runs the task synchronously on the TaskScheduler provided
Start()	Starts the task scheduling it for execution on the current TaskScheduler
Start(TaskScheduler)	Starts the task scheduling it for execution on the provided TaskScheduler
Wait()	Waits for the task to complete execution
Wait(TimeSpan)	Waits for the task to complete execution for the duration of TimeSpan

The System.Threading.Tasks.TaskScheduler class handle the low-level details of scheduling tasks on thread pools. The default task scheduler is based on the .NET 4 ThreadPool, which provides work-stealing for load balancing and thread injection/retirement for maximum performance, which is suitable for most applications. (You can find out more about task schedulers on MSDN at http://msdn.microsoft.com/en-us/library/dd997402%28VS.100%29.aspx.)

Nesting complex tasks

You can also nest Task objects inside of one another to complete more complex tasks. Nesting is useful for providing application programming interfaces (APIs) that execute multiple tasks, aggregate the results, and return a Task object that the caller can use to wait for completion or get notifications when the task has completed.

One goal is to have only a single function called when the user presses a button that makes the request to www.google.com for weather data, gets the response in XML format, and parses the XML into useful structures for representing the weather data. The calling function only cares when all of that activity is completed so that it can issue a call to update the display with the new data. The TPL makes breaking these operations into separate tasks easy. The following listing illustrates the nesting of `Task` objects:

C#:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Net;
using System.IO;
using System.Xml;
using System.Windows.Forms;
using System.Threading;
using System.Threading.Tasks;

namespace WeatherWatcher
{
    public partial class Form1 : Form
    {
        private string _Url = "http://www.google.com/ig/api?weather=";
        private string _BaseUrl = "http://www.google.com";

        private CurrentConditions _CurrentConditions = new CurrentConditions();
        private List<ForecastConditions> _ForecastConditions = new
    List<ForecastConditions>();
        private ForecastInformation _ForecastInformation = new
    ForecastInformation();

        public Form1()
        {
            InitializeComponent();
        }


        private Task<string> LoadForecastDataAsync(string url)
        {
            HttpWebRequest request =
                (HttpWebRequest)WebRequest.Create(url);

            Task<WebResponse> task =
                Task<WebResponse>.Factory.FromAsync(request.BeginGetResponse,
                                                    request.EndGetResponse,
                                                    null);

            return (task.ContinueWith((x) =>
                {
                    HttpWebResponse response = (HttpWebResponse)x.Result;

                    Encoding utf8Encoding = Encoding.UTF8;

                    StreamReader responseStream = new StreamReader(response.
    GetResponseStream(), utf8Encoding);
                    string content = responseStream.ReadToEnd();

                    return (content);
                }
            ));
    }


    private void BuildForecastFromXml(string xmlContent)
    {
        XmlDocument xmlDoc = new XmlDocument();

        xmlDoc.LoadXml(xmlContent);

        XmlNodeList forecastNodes = xmlDoc.GetElementsByTagName("forecast_
information");

        foreach (XmlNode node in forecastNodes)
        {
            foreach (XmlNode forecastNode in node.ChildNodes)
            {
                XmlAttributeCollection attrs = forecastNode.Attributes;

                if (forecastNode.Name == "city")
                {
                    this._ForecastInformation.City = attrs.
GetNamedItem("data").InnerText;
                }
                else if (forecastNode.Name == "postal_code")
                {
                    this._ForecastInformation.PostalCode = attrs.
GetNamedItem("data").InnerText;
                }
                else if (forecastNode.Name == "forecast_date")
                {
                    this._ForecastInformation.ForecastDate = attrs.
GetNamedItem("data").InnerText;
                }
                else if (forecastNode.Name == "current_date_time")
                {
                    this._ForecastInformation.CurrentDateTime = attrs.
GetNamedItem("data").InnerText;
                }
            }

        }

        XmlNodeList currentConditionsNodes = xmlDoc.
GetElementsByTagName("current_conditions");

        foreach (XmlNode node in currentConditionsNodes)
        {
            foreach (XmlNode currentNode in node.ChildNodes)
            {
                XmlAttributeCollection attrs = currentNode.Attributes;

                if (currentNode.Name == "condition")
                {
                    this._CurrentConditions.ConditionData = attrs.
GetNamedItem("data").InnerText;
                }
                else if (currentNode.Name == "temp_f")
                {
                    this._CurrentConditions.TempF = attrs.
GetNamedItem("data").InnerText;
                }
                else if (currentNode.Name == "temp_c")
                {
                    this._CurrentConditions.TempC = attrs.
GetNamedItem("data").InnerText;
                }
                else if (currentNode.Name == "humidity")
                {
                    this._CurrentConditions.HumidityData = attrs.
GetNamedItem("data").InnerText;
                }
                else if (currentNode.Name == "wind_condition")
                {
                    this._CurrentConditions.WindData = attrs.
GetNamedItem("data").InnerText;
                }
                else if (currentNode.Name == "icon")
                {
                    this._CurrentConditions.IconData = attrs.
GetNamedItem("data").InnerText;
                }

            }
        }

        XmlNodeList forecastDataNodes = xmlDoc.
GetElementsByTagName("forecast_conditions");

        foreach (XmlNode node in forecastDataNodes)
        {
            ForecastConditions cond = new ForecastConditions();

            foreach (XmlNode forecastDataNode in node.ChildNodes)
            {
                XmlAttributeCollection attrs = forecastDataNode.Attributes;

                if (forecastDataNode.Name == "day_of_week")
                {
                    cond.DayOfWeek = attrs.GetNamedItem("data").InnerText;
                }
                else if (forecastDataNode.Name == "low")
                {
                    cond.LowTemp = attrs.GetNamedItem("data").InnerText;
                }
                else if (forecastDataNode.Name == "high")
                {
                    cond.HighTemp = attrs.GetNamedItem("data").InnerText;
                }
                else if (forecastDataNode.Name == "icon")
                {
                    cond.IconData = attrs.GetNamedItem("data").InnerText;
                }
                else if (forecastDataNode.Name == "condition")
                {
                    cond.ConditionData = attrs.GetNamedItem("data").
InnerText;
                }
            }

            this._ForecastConditions.Add(cond);
        }

    }

    private void UpdateDisplay()
    {
        try
        {
            this._LabelForecastCity.Text = this._ForecastInformation.City + "
" + this._ForecastInformation.PostalCode;
            this._LabelCurrentDateTime.Text = this._ForecastInformation.
CurrentDateTime;
            this._PictureBoxCurrent.ImageLocation = _BaseUrl + this._
CurrentConditions.IconData;

            this._LabelCurMax.Text = "Temp: " + this._CurrentConditions.TempF
+ "/" + this._CurrentConditions.TempC;
            this._LabelCurrentMin.Text = this._CurrentConditions.
HumidityData;

            this._LabelDay1.Text = this._ForecastConditions[0].DayOfWeek;
            this._PictureBoxDay1.ImageLocation = this._BaseUrl + this._
ForecastConditions[0].IconData;
            this._LabelDay1MinMax.Text = this._ForecastConditions[0].HighTemp
+ "/" + this._ForecastConditions[0].LowTemp;

            this._LabelDay2.Text = this._ForecastConditions[1].DayOfWeek;
            this._PictureBoxDay2.ImageLocation = this._BaseUrl + this._
ForecastConditions[1].IconData;
            this._LabelDay2MinMax.Text = this._ForecastConditions[1].HighTemp
+ "/" + this._ForecastConditions[1].LowTemp;

            this._LabelDay3.Text = this._ForecastConditions[2].DayOfWeek;
            this._PictureBoxDay3.ImageLocation = this._BaseUrl + this._
ForecastConditions[2].IconData;
            this._LabelDay3MinMax.Text = this._ForecastConditions[2].HighTemp
+ "/" + this._ForecastConditions[2].LowTemp;

            this._LabelDay4.Text = this._ForecastConditions[3].DayOfWeek;
            this._PictureBoxDay4.ImageLocation = this._BaseUrl + this._
ForecastConditions[3].IconData;
            this._LabelDay4MinMax.Text = this._ForecastConditions[3].HighTemp
+ "/" + this._ForecastConditions[3].LowTemp;
        }
        catch (System.Exception ex)
        {
            System.Diagnostics.Debug.WriteLine(ex.Message);
        }
    }


    private void Form1_Load(object sender, EventArgs e)
    {
        string actualUrl = _Url + this._TextBoxLocation.Text;

        Task task = UpdateWeatherData(actualUrl);

        task.ContinueWith((x) =>
            {
                this.BeginInvoke((Action)(() => this.UpdateDisplay()));
            }
        );
    }

    private Task UpdateWeatherData(string url)
    {
            Task<string> innerTask = LoadForecastDataAsync(url);


            return (innerTask.ContinueWith((content) =>
                {
                    BuildForecastFromXml(content.Result);
                }
            )
            );
        }

        private void _ButtonGetWeatherData_Click(object sender, EventArgs e)
        {
            string actualUrl = _Url + this._TextBoxLocation.Text;

            this._ForecastConditions.Clear();

            Task task = UpdateWeatherData(actualUrl);

            task.ContinueWith((x) =>
            {
                this.BeginInvoke((Action)(() => this.UpdateDisplay()));
            }
            );

        }
    }
}
```

VB:

```
Imports System
Imports System.IO
Imports System.Net
Imports System.Text
Imports System.Xml
Imports System.Threading
Imports System.Threading.Tasks

Delegate Sub UpdateDisplayDelegate()

Public Class Form1
    Dim _Url As String = New String("http://www.google.com/ig/api?weather=")
    Dim _BaseUrl As String = New String("http://www.google.com")

    Dim _ForecastInformation As ForecastInformation = New ForecastInformation()
    Dim _CurrentConditions As CurrentConditions = New CurrentConditions()
    Dim _ForecastConditions As List(Of ForecastConditions) = New List(Of
     ForecastConditions)()

    Private Sub _ButtonGetWeatherData_Click(ByVal sender As System.Object, ByVal
     e As System.EventArgs) Handles _ButtonGetWeatherData.Click

        Dim actualUrl As String = Me._Url & Me._TextBoxLocation.Text

        Me._ForecastConditions.Clear()

        Dim myTask As Task = UpdateWeatherData(actualUrl)

        myTask.ContinueWith(Sub(x As Task)
                                Me.BeginInvoke(New
     UpdateDisplayDelegate(AddressOf UpdateDisplay))
```

```
                        End Sub)

End Sub

Private Function UpdateWeatherData(ByVal url As String) As Task
    Dim innerTask As Task(Of String) = LoadForecastDataAsync(url)

    Return (innerTask.ContinueWith(Sub(content As Task(Of String))
                                       BuildForecastFromXml(content.Result)
                                   End Sub
                                  )
           )
End Function

Private Function LoadForecastDataAsync(ByVal url As String) As Task(Of
String)
    Dim request As HttpWebRequest = WebRequest.Create(url)
    Dim myTask As Task(Of WebResponse) = Task(Of WebResponse).Factory.
FromAsync(AddressOf request.BeginGetResponse,

AddressOf request.EndGetResponse,

Nothing)

    Return (myTask.ContinueWith(Function(x As Task(Of WebResponse))
                                    Dim response As HttpWebResponse =
x.Result
                                    Dim utf8Encoding As Encoding = Encoding.
UTF8

                                    Dim responseStream As StreamReader = New
StreamReader(response.GetResponseStream(), utf8Encoding)
                                    Dim content As String = responseStream.
ReadToEnd()

                                    Return (content)

                                End Function)
            )

End Function


Private Sub UpdateDisplay()
    Me._LabelForecastCity.Text = Me._ForecastInformation.City + " " + Me._
ForecastInformation.PostalCode
    Me._LabelCurrentDateTime.Text = Me._ForecastInformation.CurrentDateTime
    Me._PictureBoxCurrent.ImageLocation = _BaseUrl + Me._CurrentConditions.
IconData

    Me._LabelCurMax.Text = "Temp: " + Me._CurrentConditions.TempF + "/" +
Me._CurrentConditions.TempC
    Me._LabelCurrentMin.Text = Me._CurrentConditions.HumidityData


    Me._LabelDay1.Text = Me._ForecastConditions.ElementAt(0).DayOfWeek
    Me._PictureBoxDay1.ImageLocation = Me._BaseUrl + Me._ForecastConditions.
ElementAt(0).IconData
    Me._LabelDay1MinMax.Text = Me._ForecastConditions.ElementAt(0).HighTemp +
"/" + Me._ForecastConditions.ElementAt(0).LowTemp

    Me._LabelDay2.Text = Me._ForecastConditions.ElementAt(1).DayOfWeek
    Me._PictureBoxDay2.ImageLocation = Me._BaseUrl + Me._ForecastConditions.
ElementAt(1).IconData
    Me._LabelDay2MinMax.Text = Me._ForecastConditions.ElementAt(1).HighTemp +
"/" + Me._ForecastConditions.ElementAt(1).LowTemp

    Me._LabelDay3.Text = Me._ForecastConditions.ElementAt(2).DayOfWeek
    Me._PictureBoxDay3.ImageLocation = Me._BaseUrl + Me._ForecastConditions.
ElementAt(2).IconData
    Me._LabelDay3MinMax.Text = Me._ForecastConditions.ElementAt(2).HighTemp +
"/" + Me._ForecastConditions.ElementAt(2).LowTemp

    Me._LabelDay4.Text = Me._ForecastConditions.ElementAt(3).DayOfWeek
    Me._PictureBoxDay4.ImageLocation = Me._BaseUrl + Me._ForecastConditions.
ElementAt(3).IconData
    Me._LabelDay4MinMax.Text = Me._ForecastConditions.ElementAt(3).HighTemp +
"/" + Me._ForecastConditions.ElementAt(3).LowTemp

End Sub

Private Sub BuildForecastFromXml(ByVal xmlContent As String)
    Dim xmlDoc As XmlDocument = New XmlDocument()

    xmlDoc.LoadXml(xmlContent)

    Dim forecastNodes As XmlNodeList = xmlDoc.GetElementsByTagName("forecast_
information")

    For Each node As XmlNode In forecastNodes
        For Each forecastNode As XmlNode In node.ChildNodes
            Dim attrs As XmlAttributeCollection = forecastNode.Attributes

            If (forecastNode.Name.Equals("city")) Then
                Me._ForecastInformation.City = attrs.GetNamedItem("data").
InnerText
            End If
            If (forecastNode.Name.Equals("postal_code")) Then
                Me._ForecastInformation.PostalCode = attrs.
GetNamedItem("data").InnerText
            End If
            If (forecastNode.Name.Equals("forecast_date")) Then
                Me._ForecastInformation.ForecastDate = attrs.
GetNamedItem("data").InnerText

            End If
            If (forecastNode.Name.Equals("current_date_time")) Then
                Me._ForecastInformation.CurrentDateTime = attrs.
GetNamedItem("data").InnerText
            End If

        Next forecastNode
    Next node

    Dim currentConditionsNodes As XmlNodeList = xmlDoc.
GetElementsByTagName("current_conditions")

    For Each node As XmlNode In currentConditionsNodes
        For Each currentNode As XmlNode In node.ChildNodes
            Dim attrs As XmlAttributeCollection = currentNode.Attributes

            If (currentNode.Name.Equals("condition")) Then
                Me._CurrentConditions.ConditionData = attrs.
GetNamedItem("data").InnerText
```

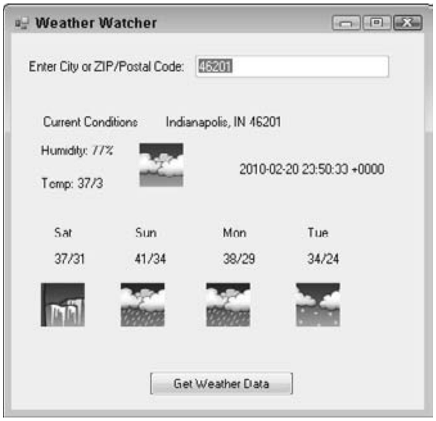
```
            End If

            If (currentNode.Name.Equals("temp_f")) Then
                Me._CurrentConditions.TempF = attrs.GetNamedItem("data").
InnerText
            End If

            If (currentNode.Name.Equals("temp_c")) Then
                Me._CurrentConditions.TempC = attrs.GetNamedItem("data").
InnerText
            End If

            If (currentNode.Name.Equals("humidity")) Then
                Me._CurrentConditions.HumidityData = attrs.
GetNamedItem("data").InnerText
            End If

            If (currentNode.Name.Equals("wind_condition")) Then
                Me._CurrentConditions.WindData = attrs.GetNamedItem("data").
InnerText
            End If

            If (currentNode.Name.Equals("icon")) Then
                Me._CurrentConditions.IconData = attrs.GetNamedItem("data").
InnerText
            End If

        Next currentNode
        Next node

    Dim forecastDataNodes As XmlNodeList = xmlDoc.
GetElementsByTagName("forecast_conditions")

    For Each node As XmlNode In forecastDataNodes
        Dim cond As ForecastConditions = New ForecastConditions()

        For Each forecastDataNode As XmlNode In node.ChildNodes

            Dim attrs As XmlAttributeCollection = forecastDataNode.Attributes

            If (forecastDataNode.Name.Equals("day_of_week")) Then
                cond.DayOfWeek = attrs.GetNamedItem("data").InnerText
            End If

            If (forecastDataNode.Name.Equals("low")) Then
                cond.LowTemp = attrs.GetNamedItem("data").InnerText
            End If

            If (forecastDataNode.Name.Equals("high")) Then
                cond.HighTemp = attrs.GetNamedItem("data").InnerText
            End If

            If (forecastDataNode.Name.Equals("icon")) Then
                cond.IconData = attrs.GetNamedItem("data").InnerText
            End If

            If (forecastDataNode.Name.Equals("condition")) Then
                cond.ConditionData = attrs.GetNamedItem("data").InnerText
            End If

        Next forecastDataNode
```

```
            Me._ForecastConditions.Add(cond)

        Next node



    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles MyBase.Load
        Dim actualUrl As String = Me._Url & Me._TextBoxLocation.Text

        Dim myTask As Task = UpdateWeatherData(actualUrl)

        myTask.ContinueWith(Sub(x As Task)
                                Me.BeginInvoke(New
    UpdateDisplayDelegate(AddressOf UpdateDisplay))
                            End Sub)

    End Sub
End Class
```

In the preceding listings, you see that the Form1_Load method and the _ButtonGetWeatherData_Click method calls the UpdateWeatherData method and gets a Task that doesn't return any data back from this method. The UpdateWeatherData method is a combination of two tasks:

- The first task executes in the LoadForecastDataAsync method and goes out to the Web to get weather data and returns a string containing the forecast data in XML format.
- The second task gets the string and calls BuildForecaseFromXml.

When this function returns, both tasks are complete, and the Task provided to the Form1_Load method or _ButtonGetWeatherData_Click method executes its continuation method, which makes a call to UpdateDisplay to reload the display with the new forecast data. Figure 10-2 shows the output of this sample application.



Enter City or ZIP/Postal Code: 46201

Current Conditions Indianapolis, IN 46201

Humidity: 77%

Temp: 37/3

2010-02-20 23:50:33 +0000

Sat 37/31

Sun 41/34

Mon 38/29

Tue 34/24

Figure 10-2: Weather watcher application using task parallel library.

Exploring Unmanaged C++ Parallel Programming with the Parallel Patterns Library (PPL)

Although much of the focus of parallel programming with Visual Studio 2010 focuses on the new TPL in the .NET 4 framework, Microsoft didn't forget about the developer of unmanaged C++ applications. Much like the TPL in the .NET 4 framework, the PPL also has a parallel for each construct to execute the same operation in parallel for a given collection of items. The PPL comes with the version of the C++ runtime installed with Visual Studio 2010. You can use this library in your C++ applications by adding the `#include <ppl.h>` directive to your code file.

The PPL is in the `Concurrency` namespace and contains a `parallel_for_each` and `parallel_for` functions for executing the same function in parallel for a given collection of items. In addition, the PPL has a `concurrent_vector` and a `concurrent_queue class`. These classes are much like the `std::vector` and `std::queue` classes found in the Standard Template Library (STL). The difference is that the `concurrent_vector and concurrent_queue` classes are thread-safe, meaning that you can safely add and remove items from these collections from multiple threads.

A major difference between the `concurrent_vector` and `std::vector` classes is that unlike the `std::vector` class, objects in the `concurrent_vector` class aren't stored in contiguous memory, so you can't randomly access items in the collection.

You can find out more about the PPL from Microsoft at `http://msdn.microsoft.com/en-us/library/dd492418%28VS.100%29.aspx`.

You can download the complete program for this example from this book's companion Web site at `www.dummies.com/go/vs2010`.

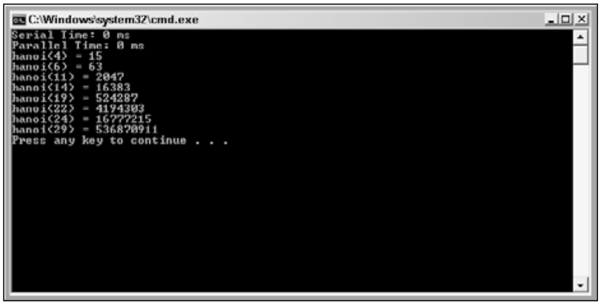
The Tower of Hanoi game example from the "Examining data parallelism in the TPL" section, earlier in this chapter, is a nice example for illustrating parallel programming in unmanaged C++. The following listing shows the Towers of Hanoi game implemented in unmanaged C++ using both serial and parallel versions for each construct.

```
#include "stdafx.h"

#include <Windows.h>
#include <ppl.h>
#include <concurrent_vector.h>
#include <array>
#include <vector>
#include <tuple>
#include <algorithm>
#include <iostream>
```

```
#include <sstream>

template <class Function>
__int64 compute_time(Function&& fn)
{
    __int64 start = GetTickCount();

    fn();

    return (GetTickCount() - start);
}

int hanoi(int n)
{
   if (n == 1)
   {
      return (1);
   }
   else
   {
      return (2 * hanoi(n - 1) + 1);
   }
}

int _tmain(int argc, _TCHAR* argv[])
{
   __int64 elapsed;

   typedef std::tr1::array<int, 8> InputArray_t;
   typedef std::tr1::tuple<int,int> ResultTuple_t;
   typedef std::vector<ResultTuple_t> SerialResults_t;
   typedef Concurrency::concurrent_vector<ResultTuple_t> ParallelResults_t;

   InputArray_t arr = { 4, 22, 19, 24, 6, 11, 29, 14};

   SerialResults_t serialResults;
   ParallelResults_t parallelResults;

   elapsed = compute_time([&]
   {
    std::for_each(arr.begin(),
                 arr.end(),
                 [&](int n)
                 {
                   serialResults.push_back(std::tr1::make_tuple(n,
                                                                hanoi(n)
                                                                )
                                         );
                 }
               );
    });

    std::cout << "Serial Time: "
              << elapsed
              << " ms"
              << std::endl;

    /*Now, compute the tower of hanoi in parallel*/

    elapsed = compute_time([&]
    {
       Concurrency::parallel_for_each(arr.begin(),
                                      arr.end(),
                                     [&](int n)
                                     {
    parallelResults.push_back(std::tr1::make_tuple(n,
                                                 parallelResults.push_
    back(std::tr1::make_tuple(n,
                                                  hanoi(n)
                                                  )
                             );

       std::sort(parallelResults.begin(),
                 parallelResults.end());
    });

    std::cout << "Parallel Time: "
              << elapsed
              << " ms"
              << std::endl;

    /*Here are the results*/
    std::for_each(parallelResults.begin(),
                  parallelResults.end(),
                  [](ResultTuple_t& pair)
                  {
                    std::cout << "hanoi("
                              << std::tr1::get<0>(pair)
                              << ") = "
                              << std::tr1::get<1>(pair)
                              << std::endl;
                  }
                 );

    return 0;
}
```

You may be thinking to yourself that the preceding syntax doesn't look like normal C++ code. The preceding code contains lambda functions that are similar to lambdas in C#. Lambdas are part of the new C++0x standard for C++ development, which is now partially supported with the Visual Studio 2010 C++ compiler. You can find out more about the new C++ standard at the MSDN blogs site at `http://blogs.msdn.com/vcblog/archive/2008/10/28/lambdas-auto-and-static-assert-c-0x-features-in-vc10-part-1.aspx`.

Figure 10-3 shows the output from the preceding sample.



```
Serial Time: 0 ms
Parallel Time: 0 ms
hanoi(4) = 15
hanoi(6) = 63
hanoi(11) = 2047
hanoi(14) = 16383
hanoi(19) = 524287
hanoi(22) = 4194303
hanoi(24) = 16777215
hanoi(29) = 536870911
Press any key to continue . . .
```

Figure 10-3: Towers of Hanoi C++ output.

Creating Parallel Queries with Parallel Language Integrated Query (PLINQ)

New with .NET 4 is a parallel version of the Language Integrated Query (LINQ) syntax introduced in .NET 3.0. PLINQ allows you to execute queries in parallel over items in collection. Book V, Chapter 7 gives more detail about LINQ.

A simple example is a query that gets a list of processes running on the system filtered by a search string. In this example, the `AsParallel` method is called on the `Process.GetProcesses` method. The `AsParallel` method tells the .NET framework to execute the query over the resulting collection in parallel. The following listing shows the C# and Visual Basic syntax.

C#:

```
var Processes = from p in System.Diagnostics.Process.GetProcesses().AsParallel()
                            where p.ProcessName.Contains(this.textBox1.Text)
                            select p;
```

VB:

```
Dim Processes = From p In Process.GetProcesses().AsParallel() _
           Where p.ProcessName.Contains(Me.TextBox1.Text) Select p
```

One pitfall of the preceding example is that because the query executes in parallel, you can't guarantee that the order is preserved. If preserving the order of the query is important, then the `AsOrdered` extension method helps you out here. The following listing shows the `AsOrdered` extension method on the query.

C#:

```
var Processes = from p in System.Diagnostics.Process.GetProcesses().AsParallel().
    AsOrdered()
                            where p.ProcessName.Contains(this.textBox1.Text)
                            select p;
```

VB:

```
Dim Processes = From p In Process.GetProcesses().AsParallel().AsOrdered() _
           Where p.ProcessName.Contains(Me.TextBox1.Text) Select p
```

You can also cancel execution of a query by using the `WithCancellation` extension method. In the following example, a task is created to wait for the user to press the *c* key to cancel a query. This task starts asynchronously. At the same time, the query to get all files ending in `.dll` that contain the letter *a* executes. For each matching file, a function that simulates a long-running operation executes, which gives the user time to cancel the query before it completes. The following listing illustrates cancelling a query.

You can download the complete program (both C# and VB versions) for this example from this book's companion Web site at www.dummies.com/go/vs2010.

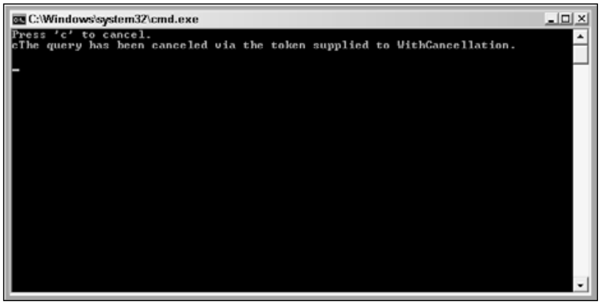
C#:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.IO;

namespace PLinqCS
{
    class Program
    {
        static void Main(string[] args)
        {
            CancellationTokenSource cs = new CancellationTokenSource();

            Task.Factory.StartNew(() =>
                {
                    ClickCancel(cs);
                });

            string[] results = null;

            try
            {
                results = (from file in Directory.GetFiles(@"C:\Windows\
    System32", "*.dll").AsParallel().WithCancellation(cs.Token)
                                    where file.Contains("a")
                                    select DoSomething(file, cs)).ToArray();
            }
            catch (OperationCanceledException e)
            {
                Console.WriteLine(e.Message);
            }
            catch (AggregateException ae)
            {
                if (ae.InnerExceptions != null)
                {
                    foreach (Exception e in ae.InnerExceptions)
                        Console.WriteLine(e.Message);
                }

            }

            if (results != null)
            {
                foreach (var v in results)
                {
                    Console.WriteLine(v);
                }
            }

            Console.WriteLine();
            Console.ReadKey();

        }

        private static string DoSomething(string file, CancellationTokenSource
    cs)
        {
            for (int i = 0; i < 5; i++)
            {
                Thread.SpinWait(100000);

                cs.Token.ThrowIfCancellationRequested();
            }

            return (file);
        }

        private static void ClickCancel(CancellationTokenSource cs)
        {
            Console.WriteLine("Press 'c' to cancel.");

            if (Console.ReadKey().KeyChar == 'c')
            {
                cs.Cancel();
            }
        }
    }
}
```

VB:

```
Imports System
Imports System.Collections
Imports System.Text
Imports System.Threading
Imports System.IO
Imports System.Threading.Tasks

Module Module1

    Sub Main()
        Dim cs As CancellationTokenSource = New CancellationTokenSource()

        Task.Factory.StartNew(Sub()
                                  ClickCancel(cs)
                              End Sub)


    Dim results As String() = Nothing

        Try
            results = (From file As String In Directory.GetFiles("C:\Windows\
    System32", "*.dll").AsParallel().WithCancellation(cs.Token)
                       Where file.Contains("a")
                       Select DoSomething(file, cs)).ToArray()

        Catch ex As OperationCanceledException
            Console.WriteLine(ex.Message)
        Catch ae As AggregateException
            If Not ae.InnerExceptions Is Nothing Then
                For Each e As Exception In ae.InnerExceptions
                    Console.WriteLine(e.Message)
                Next e
            End If
        End Try

        If Not results Is Nothing Then
            For Each s As String In results
                Console.WriteLine(s)
            Next s

        End If

        Console.WriteLine()
        Console.ReadKey()


    End Sub

    Private Function DoSomething(ByVal file As String, ByRef cs As
    CancellationTokenSource) As String
        For i As Integer = 0 To 5 Step 1
            Thread.SpinWait(100000)
            cs.Token.ThrowIfCancellationRequested()
        Next i

        Return (file)
    End Function

    Private Sub ClickCancel(ByRef cs As CancellationTokenSource)
        Console.WriteLine("Press 'c' to cancel.")

        If (Console.ReadKey().KeyChar = "c") Then
            cs.Cancel()
        End If


    End Sub
End Module
```

Figure 10-4 shows the result of cancelling the query.



```
Press 'c' to cancel.
cThe query has been canceled via the token supplied to WithCancellation.

_
```

Figure 10-4: Cancelling a PLINQ query.

Chapter 11: Writing Silverlight Applications

In This Chapter

- **Exploring Silverlight for Visual Studio 2010**
- **Adding content to your Web application**
- **Enhancing your application with interactive content**

Microsoft Silverlight is the latest platform for developing visually rich content for Web applications. Silverlight integrates multimedia, graphics, animation, and interactivity all into one cross-platform and cross-platform runtime environment. Search engines can index and search Silverlight content because Silverlight applications aren't compiled, but are declared in the Extensible Application Markup Language (XAML) format files. These XAML files are interpreted by the Silverlight runtime when executed by the browser. Starting with Silverlight 2, you can develop Silverlight applications using any .NET language. This feature makes it easy for developers to apply their .NET framework development skills to develop enhanced Web applications with Silverlight. You can also develop Silverlight applications with dynamic scripting languages, such as Ruby and Python.

Silverlight has also become the tool of choice for delivering multimedia content on the Web. Silverlight supports all the major audio and video formats and appears in a majority of Web sites that deliver radio, television shows, and even movies. With Silverlight 3, you can even deliver high-definition video over the Web. Silverlight lets you watch your favorite television episodes that you missed or try out new shows when it's most convenient for you.

Getting Started with Silverlight

At the time of this writing, Silverlight 3 is the latest version of the Silverlight platform. Silverlight is by far the richest platform for developing the next generation of Web applications. The power of Silverlight is that it leverages the .NET platform and uses the programming languages of .NET making Web development easily accessible for programmers who have only developed desktop applications. Silverlight is making the Web the medium of choice for delivering informational, educational, and entertainment content to people all around the world. The next generation of Web applications is currently being developed with Silverlight.

To develop Silverlight applications in Visual Studio 2010, you need to install the Silverlight Software Development Kit (SDK), the developer runtime, and the Visual Studio project templates. You can install all these using the Microsoft Web Platform. You can find all you need at http://silverlight.net/getstarted. The Web Platform installer automatically detects the installation of required prerequisites and installs those components that you need.

Creating a Silverlight application

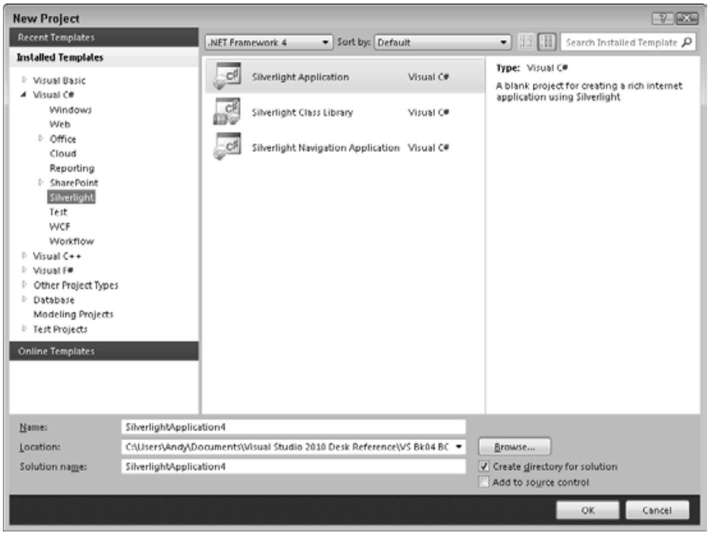
After you have the necessary components for Silverlight, you can create Silverlight applications. To create a Silverlight application in Visual Studio 2010:

1. **In Visual Studio 2010, choose File➪New➪Project from the main menu.**

   The New Project dialog box appears.

2. **Select Silverlight from the Installed Templates pane under Visual Basic or Visual C# application types.**

   The list of Silverlight project templates appears in the middle pane of the New Project dialog box, as shown in Figure 11-1.


New Project

.NET Framework 4
Sort by: Default

Silverlight Application	Visual C#
Silverlight Class Library	Visual C#
Silverlight Navigation Application	Visual C#

Type: Visual C#

A blank project for creating a rich internet application using Silverlight

Name: SilverlightApplication4

Location: C:\Users\Andy\Documents\Visual Studio 2010 Desk Reference\VS Bk04 BC

Solution name: SilverlightApplication4

☑ Create directory for solution

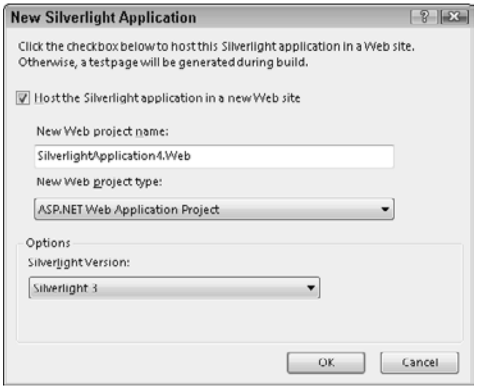
□ Add to source control

Figure 11-1: The New Project dialog contains the Silverlight project templates.

3. **Select Silverlight Application and click OK.**

   The New Silverlight Application dialog box appears, as shown in Figure 11-2.

4. **Choose the type of Web application project and Silverlight version and click OK.**



New Silverlight Application

Click the checkbox below to host this Silverlight application in a Web site. Otherwise, a test page will be generated during build.

☑ Host the Silverlight application in a new Web site

New Web project name: SilverlightApplication4.Web

New Web project type: ASP.NET Web Application Project

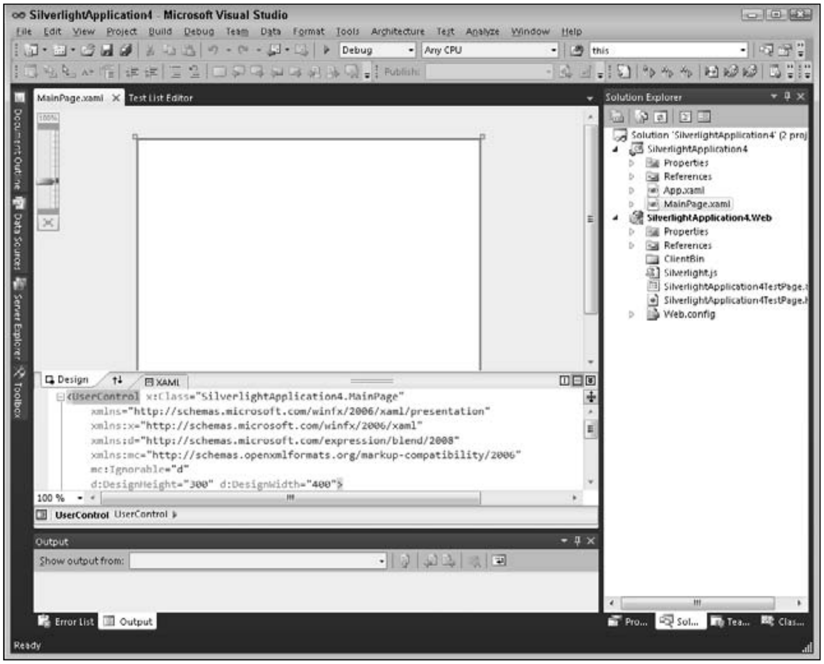
Options

Silverlight Version: Silverlight 3

Figure 11-2: The new Silverlight application configures Silverlight application settings.

Visual Studio creates a Silverlight application with the main form visible in both Design view and Source view where the XAML code for the main form is visible, as shown in Figure 11-3.

Extensible Application Markup Language (XAML) is a file format that is used to declaratively create user interfaces without the use of a compiled computer language like C#. The compiler interprets the XAML file and creates the necessary code to display the user interface elements when the application executes. Using XAML allows even non-developers to create and modify user interfaces easily without writing and compiling code.

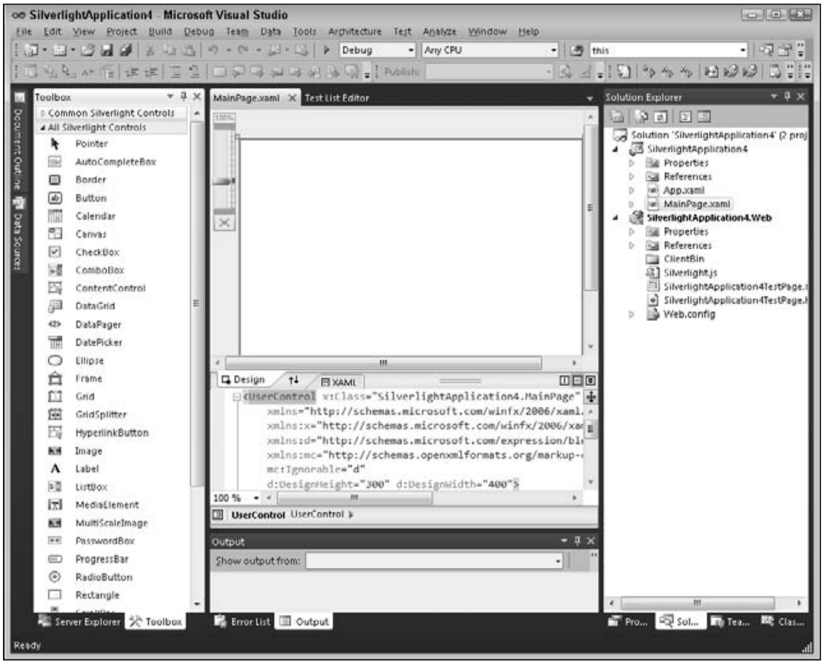


```
<UserControl x:Class="SilverlightApplication4.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">
```

SilverlightApplication4 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Format Tools Architecture Test Analyze Window Help
Debug
Any CPU
MainPage.xaml
Test List Editor
Solution Explorer
Solution 'SilverlightApplication4' (2 proj
SilverlightApplication4
Properties
References
App.xaml
MainPage.xaml
SilverlightApplication4.Web
Properties
References
ClientBin
Silverlight.js
SilverlightApplication4TestPage.
SilverlightApplication4TestPage.
Web.config
Design
XAML
100 %
UserControl UserControl
Output
Show output from:
Error List
Output
Ready

Figure 11-3: Visual Studio shows the main form in the design view and source view.

Visual Studio loads the controls you can use in your application in the Toolbox, as shown in Figure 11-4.



```
<UserControl x:Class="SilverlightApplication4.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml.
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xa
    xmlns:d="http://schemas.microsoft.com/expression/bl
    xmlns:mc="http://schemas.openxmlformats.org/markup-
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">
```

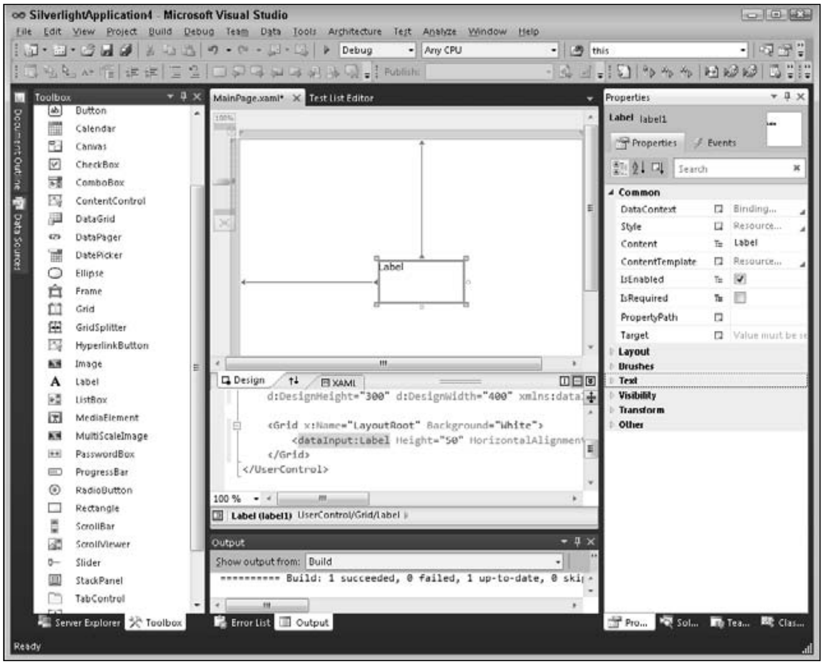
Figure 11-4: Visual Studio adds controls to the Toolbox for Silverlight applications.

Adding controls to your application

You can drag controls from the Toolbox onto the main form in the design view. To start, drag a `Label` control from the Toolbox onto the main form. You see the label in the design editor and XAML view, as shown in Figure 11-5.

You can change the text of the label to read "Hello, Silverlight World!". To change the text, select the `Content` property in the Property Explorer, as shown in Figure 11-6.

Click Ctrl+F5 or Debug➪Start Without Debugging to run the application. Visual Studio starts the Web application in your browser. Figure 11-7 shows "Hello, Silverlight World!" in your main form.



```
    d:DesignHeight="300" d:DesignWidth="400" xmlns:data

    <Grid x:Name="LayoutRoot" Background="White">
        <dataInput:Label Height="50" HorizontalAlignment
    </Grid>
</UserControl>
```

Label (label1) UserControl/Grid/label

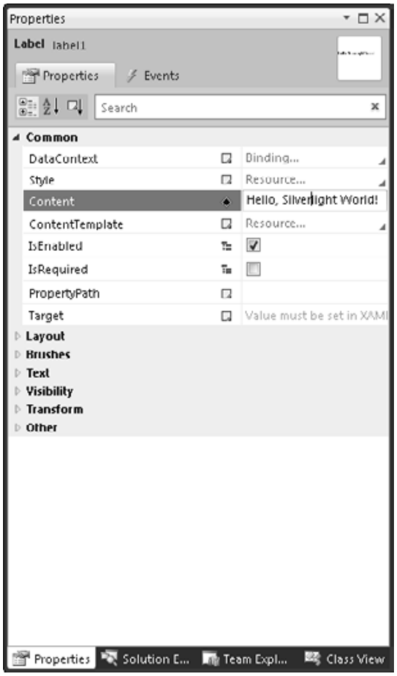
Show output from: Build

```
========== Build: 1 succeeded, 0 failed, 1 up-to-date, 0 ski
```

Label label1

DataContext: Binding...
Style: Resource...
Content: Label
ContentTemplate: Resource...
IsEnabled: ☑
IsRequired: □
PropertyPath:
Target: Value must be se

Figure 11-5: The Label appears in the Design and XAML views.



Label label1

DataContext: Binding...
Style: Resource...
Content: Hello, Silverlight World!
ContentTemplate: Resource...
IsEnabled: ☑
IsRequired: □
PropertyPath:
Target: Value must be set in XAML

Figure 11-6: Change the label's text in the Property Explorer.

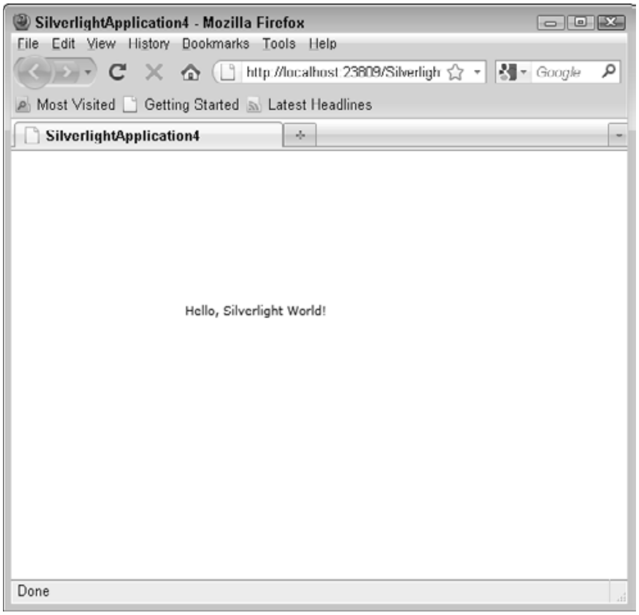

SilverlightApplication4 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:23809/Silverligh
Google
Most Visited Getting Started Latest Headlines
SilverlightApplication4
Hello, Silverlight World!
Done

Figure 11-7: The Silverlight application runs in your Web browser.

Creating Content for Web Applications

You can download the complete program (both C# and VB versions) for this example from this book's companion Web site at `www.dummies.com/go/vs2010`.

You can use the tools provided in Silverlight to create content for your Web application. In this example, you create a simple page that uses the Silverlight controls to display a simple digital clock. Start by creating a Silverlight application. (See the section "Creating a Silverlight application," earlier in this chapter.)

To create a digital clock:

1. **Drag a Canvas from the Toolbox onto your main form in the Design view.**

   A blank canvas appears in the form in the Design view.

2. **Drag a rectangle from the Toolbox inside the Canvas.**

   A Rectangle object with sizing handles appears inside the Canvas.

3. **Using the sizing handles, make the Rectangle completely cover the Canvas.**

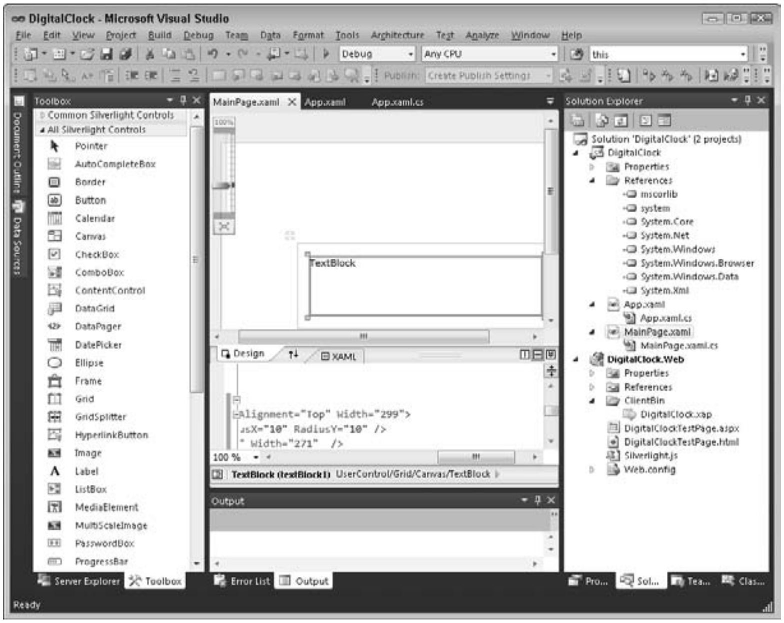
   The Rectangle now covers the entire Canvas.

4. **Drag a `TextBlock` from the Toolbox to the interior of the Rectangle.**

   A `TextBlock` with sizing handles appears in the interior of the Rectangle.

5. **Use the sizing handles to make the `TextBlock` nearly the same size as the Rectangle.**

   The `TextBlock` appears in the Rectangle, as shown in Figure 11-8.



```
Alignment="Top" Width="299">
usX="10" RadiusY="10" />
" Width="271"  />
```

TextBlock (textBlock1) UserControl/Grid/Canvas/TextBlock

Figure 11-8: The clock outline appears in the Design pane of Visual Studio.

Adding style to your clock

What appears so far is just the outline of the digital clock. You can make this outline look more like a clock, perhaps with some visual depth, a border, and maybe rounded corners, thanks to the `Style` property. You can create styles with XAML and then apply those styles to controls. You can place styles in any `.xaml` file. In this case, it would be nice for these styles to have scope in the entire application, so putting them in the `App.xaml` is a good choice.

TIP

You don't need to create styles separately in XAML. You can create the same effects by setting the appropriate properties on the elements themselves. Nevertheless, using separate styles is preferred. If you have more than one element that you want to have the same appearance, then you can easily change the style and all elements that use that style automatically change. Using separate styles makes for more flexible and maintainable code.

First, you want to style the Rectangle element that forms the background of the clock. This style gives the Rectangle element rounded corners and uses linear gradients to draw a border and fill the Rectangle element. The Stroke property controls the outline of the Rectangle. The Fill property sets the brush that fills the Rectangle element, and the StrokeThickness property determines the thickness of the border. For the digital clock, you use a LinearGradiantBrush for the border and fill of the Rectangle. The following style sets all these properties:

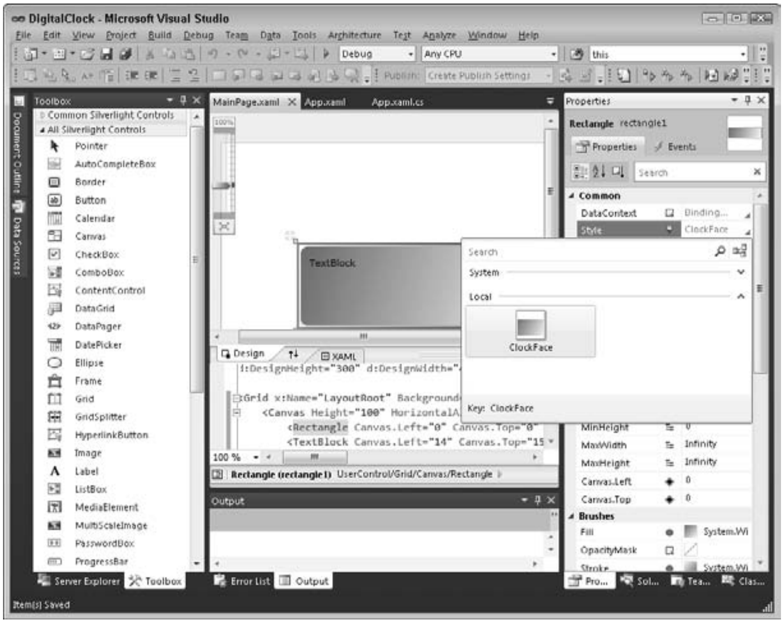
```
<Application xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             x:Class="DigitalClock.App"
             >
    <Application.Resources>

        <Style x:Key="ClockFace" TargetType="Rectangle">
            <Setter Property="StrokeThickness" Value="4"></Setter>
            <Setter Property="Fill">
                <Setter.Value>
                    <LinearGradientBrush>
                        <GradientStop Offset="0.0" Color="DodgerBlue" />
                        <GradientStop Offset="1.0" Color="White" />
                    </LinearGradientBrush>
                </Setter.Value>
            </Setter>
            <Setter Property="Stroke">
                <Setter.Value>
                    <LinearGradientBrush>
                        <GradientStop Offset="0.0" Color="White" />
                        <GradientStop Offset="1.0" Color="DodgerBlue" />
                    </LinearGradientBrush>
                </Setter.Value>
            </Setter>
        </Style>

    </Application.Resources>
</Application>
```

After you create this style, you can apply it to the Rectangle element by selecting the Rectangle in the Design view using the Properties window, as shown in Figure 11-9.

Jazzing up your clock's text

Now, you want to create a custom style for the clock's text. You can simply add a style to App.xaml by creating yellow text with a font size of 36.

```
<Style x:Key="ClockText" TargetType="TextBlock">
            <Setter Property="FontSize" Value="36" />
            <Setter Property="Foreground">
                <Setter.Value>
                    <SolidColorBrush Color="Yellow">
                    </SolidColorBrush>
                </Setter.Value>
            </Setter>
        </Style>
```



```
d:DesignHeight="300" d:DesignWidth="4
<Grid x:Name="LayoutRoot" Background
    <Canvas Height="100" HorizontalA
        <Rectangle Canvas.Left="0" Canvas.Top="0"
        <TextBlock Canvas.Left="14" Canvas.Top="15
```

100 %

Rectangle (rectangle1) UserControl/Grid/Canvas/Rectangle

Properties

Rectangle rectangle1

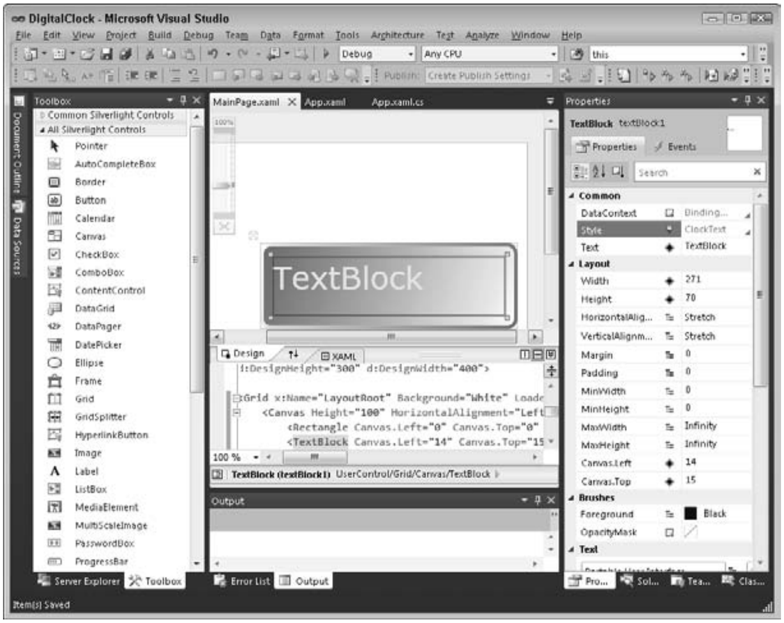
Common
DataContext Binding...
Style ClockFace

Key: ClockFace

MinHeight 0
MaxWidth Infinity
MaxHeight Infinity
Canvas.Left 0
Canvas.Top 0
Brushes
Fill System.Wi
OpacityMask
Stroke System.Wi

Figure 11-9: Applying the `Clock Face` style gives the Rectangle element visual depth.

Now you can go to the Design view of `MainPage.xaml` and apply the `Clock Text` to the `TextBlock` by selecting it and assigning the `ClockText` style to the `Style` property in the Properties window. Figure 11-10 shows the result.



```
d:DesignHeight="300" d:DesignWidth="400">
<Grid x:Name="LayoutRoot" Background="White" Loade
    <Canvas Height="100" HorizontalAlignment="Left
        <Rectangle Canvas.Left="0" Canvas.Top="0"
        <TextBlock Canvas.Left="14" Canvas.Top="15
```

100 %

TextBlock (textBlock1) UserControl/Grid/Canvas/TextBlock

Properties

TextBlock textBlock1

Common
DataContext Binding...
Style ClockText
Text TextBlock
Layout
Width 271
Height 70
HorizontalAlig... Stretch
VerticalAlignm... Stretch
Margin 0
Padding 0
MinWidth 0
MinHeight 0
MaxWidth Infinity
MaxHeight Infinity
Canvas.Left 14
Canvas.Top 15
Brushes
Foreground Black
OpacityMask
Text

Figure 11-10: Adding styles to Silverlight elements gives them a customized appearance.

Displaying the current time

The layout for the digital clock isn't quite complete. The text should display the current time in a digital clock format. And in order to function like a clock, the text must change as the time advances. Now is the time to go to code to finish up the clock.

First, you need a timer that ticks every second so that you can update the clock text. The `System.Windows.Threading.DispatcherTimer` provides a timer for use in Silverlight applications. To update the clock text, you set the `Text` property of the TextBlock element when the `Tick` event of the timer occurs:

C#:

```
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using System.Windows.Threading;
using System.Text;

namespace DigitalClock
{
    public partial class MainPage : UserControl
    {
        private DispatcherTimer _Timer;

        public MainPage()
        {
            InitializeComponent();

            _Timer = new DispatcherTimer();

            _Timer.Tick += new EventHandler(_Timer_Tick);
            _Timer.Interval = new TimeSpan(0, 0, 1);
        }


        void _Timer_Tick(object sender, EventArgs e)
        {
            System.DateTime currentTime = System.DateTime.Now;

            this.textBlock1.Text = currentTime.ToString("HH:mm:ss");
        }

        private void LayoutRoot_Loaded(object sender, RoutedEventArgs e)
        {
            System.DateTime currentTime = System.DateTime.Now;

            this.textBlock1.Text = currentTime.ToString("HH:mm:ss");
            this._Timer.Start();
        }
    }


}
```

VB:

```
Imports System
Imports System.Windows
Imports System.Windows.Threading
Imports System.Windows.Controls
Imports System.Windows.Documents
Imports System.Windows.Input
Imports System.Windows.Media
Imports System.Windows.Media.Animation

Partial Public Class MainPage
    Inherits UserControl

    Private _Timer As System.Windows.Threading.DispatcherTimer

    Public Sub New()
        InitializeComponent()

        _Timer = New System.Windows.Threading.DispatcherTimer()
        _Timer.Interval = New TimeSpan(0, 0, 1)

    End Sub

    Private Sub LayoutRoot_Loaded(ByVal sender As System.Object, ByVal e As
    System.Windows.RoutedEventArgs) Handles LayoutRoot.Loaded

        Dim currentTime As System.DateTime = System.DateTime.Now

        Me.textBlock1.Text = currentTime.ToString("HH:mm:ss")

        AddHandler _Timer.Tick, AddressOf _Timer_Tick

        _Timer.Start()

    End Sub

    Private Sub _Timer_Tick(ByVal sender As Object, ByVal e As System.EventArgs)
        Dim currentTime As DateTime = System.DateTime.Now

        Me.textBlock1.Text = currentTime.ToString("HH:mm:ss")

    End Sub
End Class
```

Centering the clock's text

Most clocks display the time centered in the clock display. Unfortunately, no property on the `TextBlock` centers the text, so you have to do a little math. After you set the `Text` property of the `TextBlock`, the `ActualWidth` and `ActualHeight` properties of the `TextBlock` element adjust to the actual height and width of the rendered text. Armed with this information, centering the text just takes a little math. You know the size of the Rectangle that contains the `TextBlock`; adding the following code centers the clock text.

C#:

```
double left = (this.rectangle1.Width - this.textBlock1.ActualWidth) / 2;
double top = (this.rectangle1.Height - this.textBlock1.ActualHeight) / 2;

this.textBlock1.SetValue(Canvas.TopProperty, top);
this.textBlock1.SetValue(Canvas.LeftProperty, left);
```

VB:

```
 Dim left As Double = (Me.rectangle1.Width - Me.textBlock1.ActualWidth) / 2
      Dim top As Double = (Me.rectangle1.Height - Me.textBlock1.ActualHeight)
   / 2

      Me.textBlock1.SetValue(Canvas.TopProperty, top)
      Me.textBlock1.SetValue(Canvas.LeftProperty, left)
```

To see the digital clock in action, press Ctrl+F5 to run the application. The application loads in your browser and displays the clock, as shown in Figure 11-11.

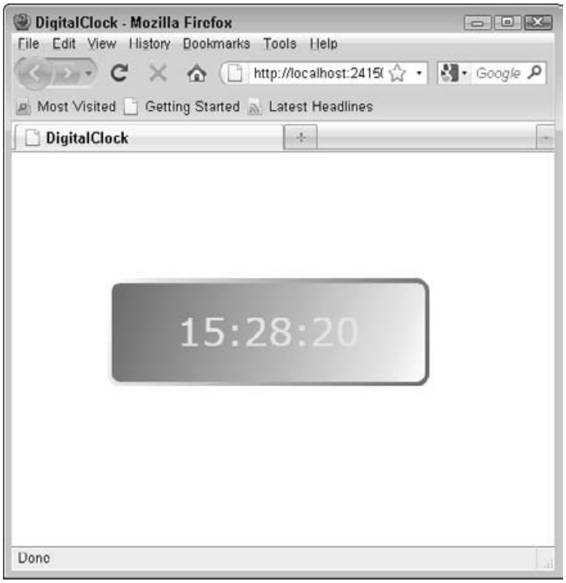


Figure 11-11: The digital clock displays in your browser.

Adding Interactive Content to Enhance the Web Experience

The Web experience has evolved from just providing content to users to one in which users can directly interact with Web sites. My children love to play educational games designed for preschoolers such as `starfall.com` and `pbskids.org`. These sites provide visually pleasing graphics and games where children can select objects and fill in shapes with colors.

Silverlight controls can also respond to mouse events just like windows in desktop applications. Using the mouse, you can move things around, draw and fill shapes, and select objects on a Web page — truly making the Web an interactive experience.

A simple example of interacting with a Silverlight application is a line drawing application. You simply click the mouse to set the start point and, while holding the left mouse button, drag the mouse to another point and release. As soon as you release the button, a line appears from the point where you clicked to the point where you released the button.

Handling mouse events in Silverlight is as simple as handling mouse events in desktop applications. The following code illustrates a simple line drawing application in Silverlight:

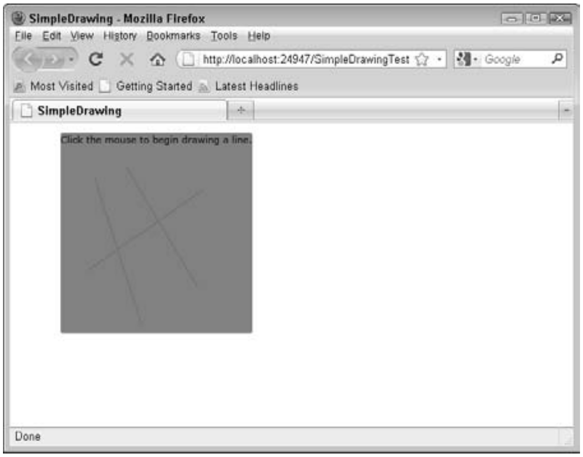
C#:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

namespace SimpleDrawing
{
    public partial class MainPage : UserControl
    {
        Point _StartPoint;
        bool _LeftButtonDown = false;

        public MainPage()
        {
            InitializeComponent();
        }

        private void LayoutRoot_MouseLeftButtonDown(object sender,
    MouseButtonEventArgs e)
        {
            _StartPoint = e.GetPosition(this);
            _LeftButtonDown = true;
        }

       private void LayoutRoot_MouseLeftButtonUp(object sender,
    MouseButtonEventArgs e)
        {
            _LeftButtonDown = false;

            Point endPoint = e.GetPosition(this);

            Line line = new Line();
            line.X1 = _StartPoint.X;
            line.X2 = endPoint.X;
            line.Y1 = _StartPoint.Y;
            line.Y2 = endPoint.Y;

            line.Stroke = new System.Windows.Media.SolidColorBrush(Color.
    FromArgb(255, 128,128,128));

            this.LayoutRoot.Children.Add(line);
        }
    }
}
```

VB:

```
Partial Public Class MainPage
    Inherits UserControl

    Private _StartPoint As Point
    Private _LeftButtonDown As Boolean = False

    Public Sub New()
        InitializeComponent()
    End Sub

    Private Sub LayoutRoot_MouseLeftButtonDown(ByVal sender As System.Object,
    ByVal e As System.Windows.Input.MouseButtonEventArgs)
        _StartPoint = e.GetPosition(Me)
        _LeftButtonDown = True
    End Sub

    Private Sub LayoutRoot_MouseLeftButtonUp(ByVal sender As System.Object, ByVal
    e As System.Windows.Input.MouseButtonEventArgs)
        _LeftButtonDown = False

        Dim endPoint As Point = e.GetPosition(Me)

        Dim line As Line = New Line()

        line.X1 = _StartPoint.X
        line.X2 = endPoint.X
        line.Y1 = _StartPoint.Y
        line.Y2 = endPoint.Y

        line.Stroke = New System.Windows.Media.SolidColorBrush(Color.
    FromArgb(255, 128, 128, 128))

        Me.LayoutRoot.Children.Add(line)
    End Sub

    Private Sub LayoutRoot_MouseMove(ByVal sender As System.Object, ByVal e As
     System.Windows.Input.MouseEventArgs)

    End Sub
End Class
```

Figure 11-12 shows the result in the browser.


SimpleDrawing - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:24947/SimpleDrawingTest
Google
Most Visited
Getting Started
Latest Headlines
SimpleDrawing
Done

Click the mouse to begin drawing a line.

Figure 11-12: Creating a simple line drawing application with Silverlight.

Chapter 12: Programming for the Cloud

In This Chapter

- **Figuring out what cloud computing is**
- **Using Visual Studio 2010 for cloud computing**
- **Creating applications for the cloud with Windows Azure**
- **Providing you application's services from the cloud**

Many small and medium-sized organizations resist purchasing the software they need to enhance their productivity for the reason that the software requires a significant investment in computer hardware and personnel to administer and deploy the software. For many organizations, these costs are prohibitive. Also, if their organization grows, your customers will need more computer hardware and more personnel to manage their software deployments.

But what if your customers could use the software they need and not have to worry about managing hardware and personnel or scaling their software upward as their organization grows? In this chapter, you discover how Microsoft's Windows Azure platform lets you create your applications and deploy them on their cloud platform freeing your customers from cost-prohibitive deployment and scalability issues. With Windows Azure, your applications execute in Microsoft's data centers, and your customers use your software across the Internet.

Reaping the Benefits of Cloud Computing

Cloud computing is all the rage in application development today. Everywhere you turn, people in the software industry are all talking about the cloud and how they can make their software ready for the cloud. But what is cloud computing?

Simply put, *cloud computing* is a way of deploying applications and data scattered throughout remote locations and accessed via the Internet. The *cloud* is a metaphor for the Internet, a vast, amorphous space across which information flows.

An old idea becomes new again: Cloud computing

Cloud computing isn't a new idea. It actually originated in the 1960s, but the enormous investments in network bandwidth capacity in the 1990s have made large-scale cloud computing feasible today. Network bandwidth capacity far outpaced demand by the end of the 1990s and was one of the contributing factors to bursting of the so-called dot-com bubble in 2000. By 2000, many companies were utilizing less than 20 percent of the available network bandwidth. The 2000s saw a rapid increase in power of computers as well as the massive delivery of media content via the Internet. Cloud computing is an idea whose time has finally arrived.

Cloud computing is a dramatic paradigm shift away from software being installed on a company's or user's computer resources. Instead, that software is installed at remote data centers potentially scattered around the world, and users access it via the Internet. Cloud computing is a move toward a service-based approach to software where third-party vendors provide the technology resources required for an application or data to be deployed. Cloud computing abstracts the details of a software application's hardware and system requirements from the user, making software much more of a service.

With cloud computing, companies no longer have to invest in technology resources to deploy an application. The application is hosted at remote locations by a cloud vendor. If the company grows and needs the software to scale upward for increased usage, the company doesn't need to worry about acquiring more hardware. The service provider takes care of all these details. Service providers offer their customers Service Level Agreements (SLAs) that promise a minimum level of service and usually come with financial penalties for not meeting those levels of service. Customers typically pay for the resources they use, so cloud computing allows smaller and mid-sized companies access to large-scale software applications without the cost-prohibitive expense and overhead of acquiring and managing a farm of servers and network resources.

Programming for the Cloud with Visual Studio

Not to be left behind in the shift to cloud computing, Microsoft introduced the Windows Azure platform for cloud computing. Windows Azure is the development, service hosting, and service management environment for applications developed on Microsoft's cloud computing platform. Windows Azure provides developers with on-demand access to host, manage, and scale web applications using Microsoft data centers. You can read more about Windows Azure at `www.microsoft.com/windowsazure`.

The Windows Azure platform is an exciting new way to develop and deploy your applications. You can also use the Azure marketplace to exhibit your Azure services to prospective customers. With Windows Azure, you don't need to invest in computer hardware and network bandwidth resources. You simply do what you do best — develop software and let your application live on the cloud as you go about developing the best software that meets market demands.

To developing applications for the cloud using Windows Azure, you need to download and install the Windows Azure tools from Microsoft at `www.microsoft.com/downloads` and then search for Azure Software Development Kit in the search box. The Windows Azure tools provide the Azure Software Development Kit (SDK) as well as tools for testing and debugging your applications locally before deploying to the cloud.

Deploying ASP.Net Web Services on the Cloud

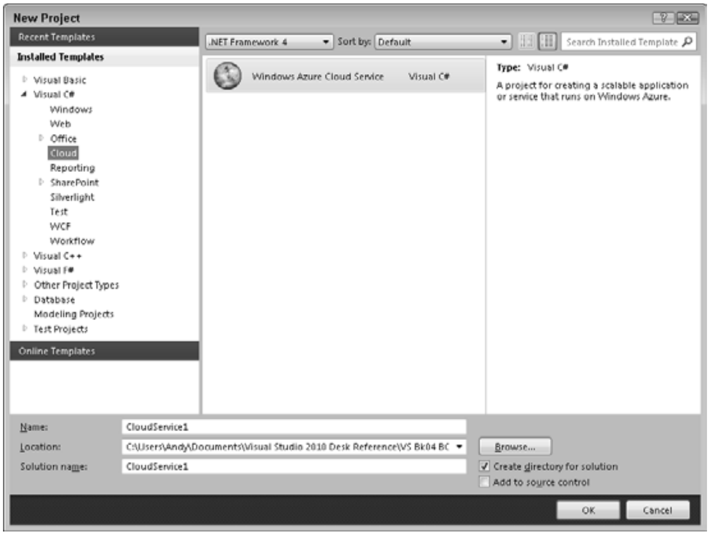
Visual Studio 2010 comes with a project template for developing applications with Windows Azure. To create an application for the cloud:

1. **Choose File⇨New⇨Project from the main menu of Visual Studio.**

   The New Project dialog box appears.

2. **Choose Cloud under the Visual C# or Visual Basic project categories.**

   The Windows Azure Cloud Service template appears in the middle pane of the New Project dialog box, as shown in Figure 12-1.



.NET Framework 4 Sort by: Default

Type: Visual C#

A project for creating a scalable application or service that runs on Windows Azure.

Name: CloudService1

Location: C:\Users\Andy\Documents\Visual Studio 2010 Desk Reference\VS Bk04 BC

Solution name: CloudService1

☑ Create directory for solution

□ Add to source control

Figure 12-1: Visual Studio has a project template for developing applications for Windows Azure.

3. **Click OK.**

   The New Cloud Service Project appears, as shown in Figure 12-2.

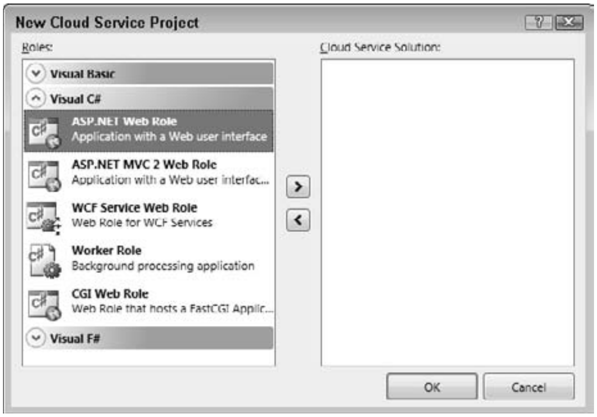

New Cloud Service Project
Roles:
Visual Basic
Visual C#
ASP.NET Web Role
Application with a Web user interface
ASP.NET MVC 2 Web Role
Application with a Web user interfac...
WCF Service Web Role
Web Role for WCF Services
Worker Role
Background processing application
CGI Web Role
Web Role that hosts a FastCGI Applic...
Visual F#
Cloud Service Solution:
OK
Cancel

Figure 12-2: Visual Studio offers various types of cloud service applications.

4. **Choose the appropriate project roles and click the right arrow to move the project role to the right panel of the New Cloud Service Project dialog box.**

   For this example, select ASP.Net Web Role to create an ASP.NET Web application.

5. **Click OK.**

   Visual Studio creates a Windows Azure solution with an ASP.NET Web role, as shown in Figure 12-3.

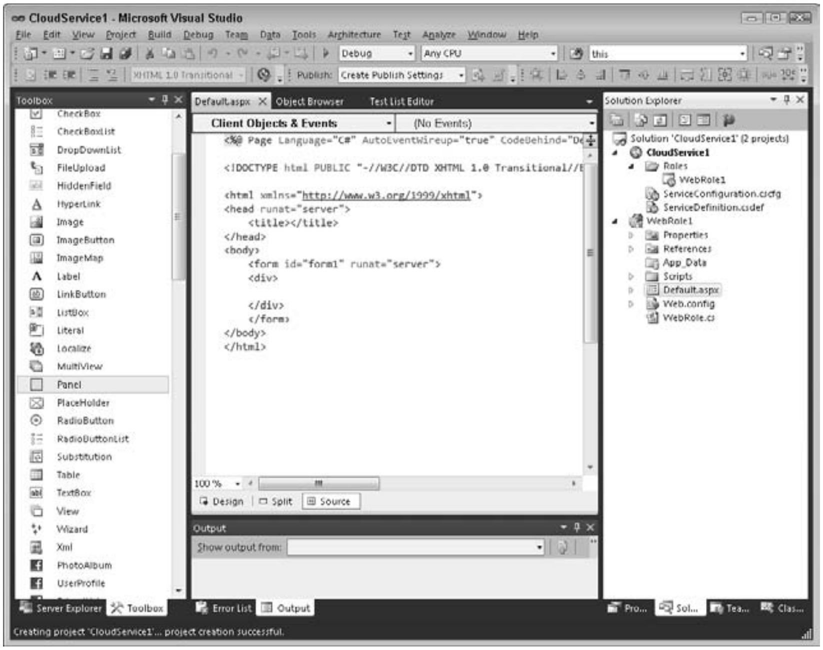
To execute a Windows Azure application from Visual Studio 2010 on your computer, you must run Visual Studio with Administrator privileges.

You must have at least SQL Server 2008 Express installed to execute Windows Azure projects on your development computer.

Choosing a Web Role

Visual Studio 2010 has five different roles for cloud service projects, each role determines how your application operates in the cloud and what type of services it can provide:

- **ASP.NET Web Role:** This role is for deploying an application with a web user interface. Use this role when deploying an ASP.NET application that has a web user interface that users will use to interact with the application. This role can have only one HTTP, one HTTPS external endpoint, plus only one HTTP internal endpoint, making this role suitable for Web applications.



```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="De

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//E

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

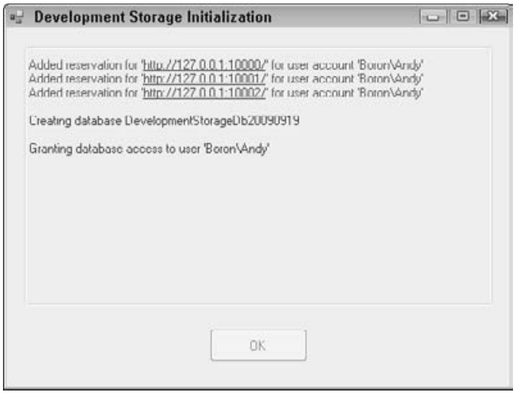
Creating project 'CloudService1'... project creation successful.

Figure 12-3: Visual Studio creates a Windows Azure solution with an ASP. Net web application.

- **ASP.NET MVC 2 Web Role:** This role is similar to the ASP.NET Web Role, but instead creates an ASP.NET application that uses the Model View Controller (MVC) assemblies for building ASP.NET applications. This role can have only one HTTP and one HTTPS external endpoints, plus only one HTTP internal endpoint.
- **WCF Service Web Role:** Use this role when deploying WCF services to the cloud.
- **Worker Role:** A worker role process performs background processing and may communicate with other services and Internet-based services.
- **F# Worker Role:** A worker role project using the F# programming language.

Adding Content to Your Application

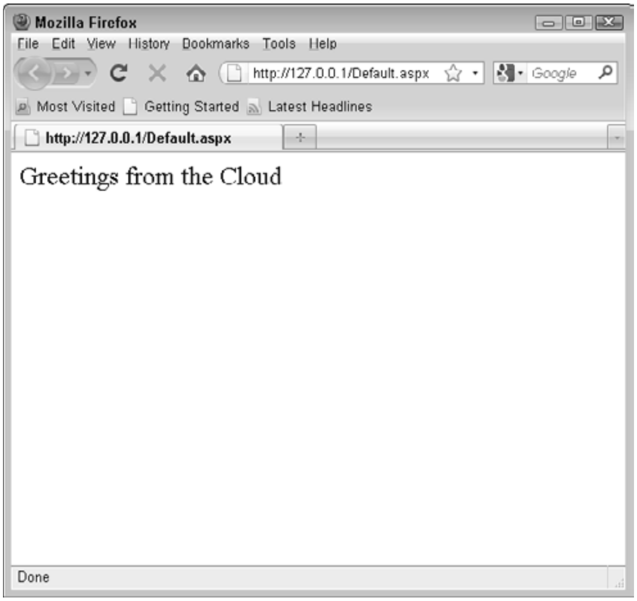
You can add content to your ASP.NET application to execute in the cloud. Drag a label onto the Design page and set the text property in the Properties panel to "Greetings from the Cloud". Click Ctrl+F5 to execute. When you first execute your application, Windows Azure displays a dialog box, similar to the one shown in Figure 12-4, that tells you it's allocating storage for your application in SQL Server.



```
Added reservation for 'http://127.0.0.1:10000/' for user account 'Boron\Andy'
Added reservation for 'http://127.0.0.1:10001/' for user account 'Boron\Andy'
Added reservation for 'http://127.0.0.1:10002/' for user account 'Boron\Andy'

Creating database DevelopmentStorageDb20090919

Granting database access to user 'Boron\Andy'
```

Figure 12-4: Windows Azure allocates storage space for your application in SQL Server 2008.

After Windows Azure has allocated storage for your application, the application executes just like any other ASP.Net application, and the results appear in your browser, as shown in Figure 12-5.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://127.0.0.1/Default.aspx
Google
Most Visited
Getting Started
Latest Headlines
http://127.0.0.1/Default.aspx
Done

Greetings from the Cloud

Figure 12-5: Windows Azure simulates the cloud and executes an ASP.Net application on your computer.

Deploying Your Applications on the Cloud

After your application is completed and tested, you're ready to deploy it on the cloud. You can deploy Windows Azure projects to the cloud using the Azure Services Development Portal.

To deploy your application on the Azure Services Development Portal:

1. **Set up an account on the Azure Services Development Portal.**

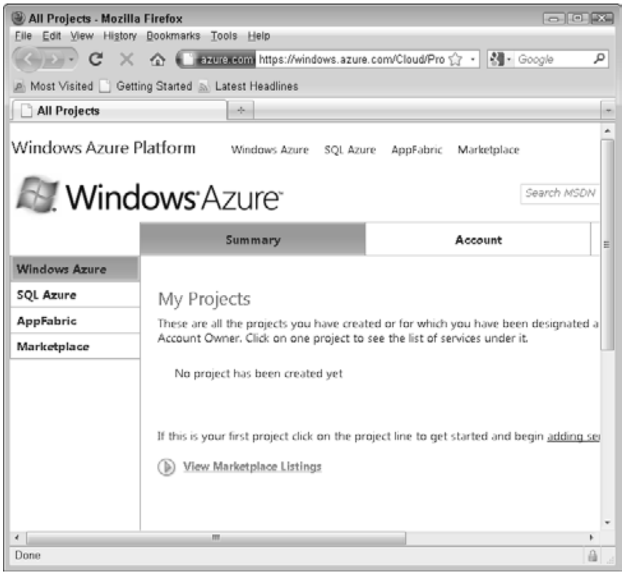
   Microsoft offers several rate plans for its Azure Services. You can find a complete list of rate plans and prices at `www.microsoft.com/windowsazure` and click the offers link at the top of the page. At the time of this writing, prices and offers are changing frequently, so be sure to check often for the latest pricing.

2. **Create a project on the developer portal.**

   You have to give the project a name and a description. The Azure Services Development Portal assigns an application Id for your project. You need this ID to deploy and publish your project from Visual Studio.

3. **In Visual Studio, right-click the cloud service in the Solution Explorer and select Browse to Portal from the context menu.**

   Visual Studio starts the browser and takes you to the Azure Services Development Portal. Figure 12-6 shows the Azure Service Development Portal. Visual Studio also navigates to your project's Publish folder, as shown in Figure 12-7.


All Projects - Mozilla Firefox
File Edit View History Bookmarks Tools Help
azure.com https://windows.azure.com/Cloud/Pro
Google
Most Visited Getting Started Latest Headlines
All Projects
Windows Azure Platform
Windows Azure SQL Azure AppFabric Marketplace
Windows Azure
Search MSDN
Summary
Account
Windows Azure
SQL Azure
AppFabric
Marketplace
View Marketplace Listings
Done

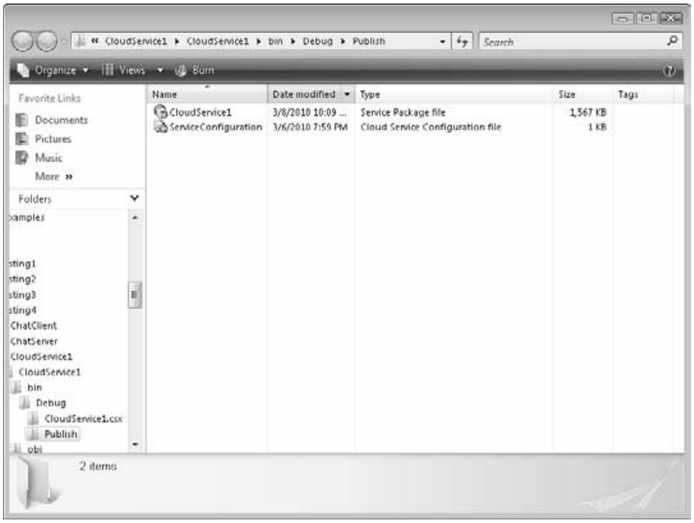
My Projects

These are all the projects you have created or for which you have been designated a Account Owner. Click on one project to see the list of services under it.

No project has been created yet

If this is your first project click on the project line to get started and begin adding se

Figure 12-6: Configure your cloud services on the Azure Services Development Portal.



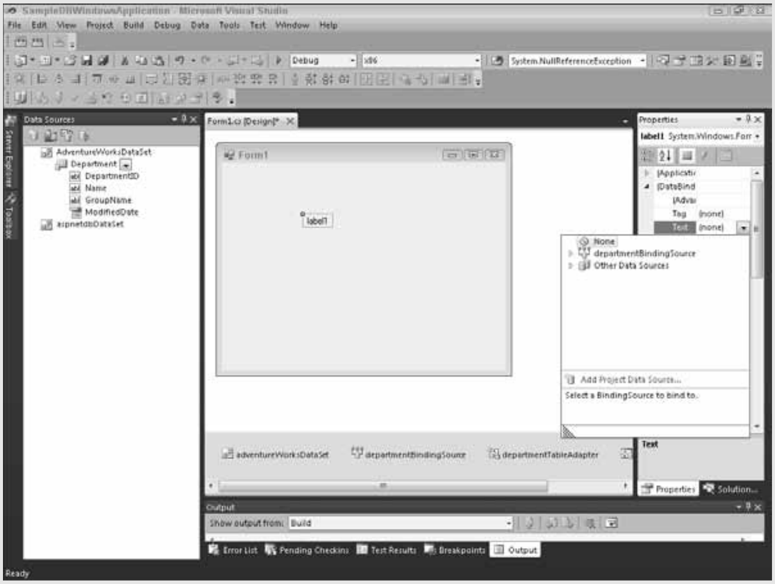
Name	Date modified	Type	Size	Tags
CloudService1	3/8/2010 10:09 ...	Service Package file	1,567 KB	
ServiceConfiguration	3/6/2010 7:59 PM	Cloud Service Configuration file	1 KB	

Figure 12-7: Visual Studio navigates to your project's publish folder.

4. **Upload your Service Page File and your Cloud Service Configuration File to the developer portal.**

Book V

Getting Acquainted with Data Access



Debug

x86

System.NullReferenceException

Tag (none)

Text (none)

Select a BindingSource to bind to.

Show output from: Build

Contents at a Glance

Chapter 1: Accessing Data with Visual Studio

In This Chapter

- **Taking a sneak peek at ADO.NET**
- **Getting acquainted with databases**
- **Adding data controls to Windows and Web Forms**

Nowadays, it's hard to find an application that isn't data-driven. Even if the application isn't centered around entering and viewing data, you might be surprised to find a tremendous amount of data access occurring behind the scenes — especially for Web sites. What might look like a static, boring Web site might be one that was built by using dynamic Web pages that get their content from a database.

Applications, such as Windows SharePoint Services, store all their information in databases. For example, when a user uploads an Excel document to a SharePoint document library, the file is stored in a database. Don't bother to traverse the file directory and look for the file, though, because it's not there.

Even when data isn't involved in an application, metadata might still be involved. *Metadata* is data about data, and it's showing up everywhere. For example, when you take a picture with a digital camera, the camera automatically stores the image resolution and thumb file format as metadata within the JPEG image. A photo gallery application, for example, may use this metadata to determine how the image is to appear.

Yes, the world has gone data crazy. Fear not, however, because Visual Studio has everything you need for accessing data and metadata in your applications. This chapter shows you how to use Visual Studio to create data-driven applications without getting bogged down in all the technology that makes data access possible.

Accessing Data with Visual Studio and .NET

Visual Studio provides many tools for accessing and manipulating data. The underlying technology that makes the tools in Visual Studio work is ADO.NET. It provides all the commands and widgets you need in order to work with data in your application. With ADO.NET, you can

- Connect to data sources.
- Retrieve data from data sources.
- Display data in your application.
- Update data to reflect changes your users make.

Meeting ADO.NET

ADO.NET is the technology in the Microsoft .NET Framework that makes it possible to work with data in your applications. You can work with ADO.NET by using wizards in Visual Studio or by writing your own code to access ADO.NET. Either way, it helps to understand conceptually how ADO.NET works.

ADO.NET is a part of the .NET Framework in the same way that ASP.NET and Windows Forms are part of the .NET Framework. The .NET Framework provides all the code that brings these technologies to life. Visual Studio 2010 provides wizards that generate code that let you use these technologies of the .NET Framework without digging into the details of how the technologies work.

Starting at the data source

Data access in ADO.NET begins with data sources. Starting with data sources makes sense: You can't display data until you know its source.

ADO.NET can access almost any data source imaginable. Using ADO.NET, you can access these data sources:

- **Databases, such as SQL Server and Oracle:** See Book V, Chapter 6 for an overview of Microsoft SQL Server.
- **XML files:** Book V, Chapter 4 shows you how to access XML files.
- **Objects:** Book V, Chapter 2 of this book shows you how to connect to an object.
- **Web services:** Book III, Chapter 6 shows you how to work with Web services.
- **Local data stored in your project:** See Book V, Chapter 6 for more information on local data sources.

Visual Studio provides two tools for creating and managing data sources:

- The **Data Source Configuration Wizard** walks you through creating a new data source.
- The **Data Sources window** is a container for managing all your project's data sources. It's available only in Windows applications.

ASP.NET includes a set of data source controls for configuring data sources, and Windows has the new `BindingSource` component for wiring data sources to data-bound controls.

Retrieving data from data sources

Setting up data sources for your projects is the first step in retrieving data from the data sources. You have two options for telling ADO.NET which data to get from data sources:

- Write code by using ADO.NET commands.
- Use the Visual Studio `TableAdapter`.

Before ADO.NET, an application stayed connected to the data source while getting data. You had to manage the process of walking through data records to reach the record you wanted. ADO.NET eliminates all that by using a *disconnected model.* With ADO.NET, you download your data from your data source and then disconnect from it. If you make changes to the data that you want to send to the data source, you have to connect to the data source again.

In a *connected model,* the data always remains in the data source. You connect to the data source, read data, and then display that data as you're reading it. You make changes to data while you're connected to the data source.

In the ADO.NET disconnected model, you don't have live access to the data to execute commands. Instead, you download a copy of the data, work with it offline, and then reconnect to the data source to pass back any changes. When you work with a disconnected model, you need a place to store the data you download from your data source. ADO.NET has two methods for executing offline downloads:

- `DataSet`**:** Stores the data you download from your data source. The `DataSet` is made up of tabular sets of data, similar to a database. A `DataSet` is essentially an offline database. You connect to the data source and download your data to the `DataSet`. The `DataSet` includes commands for accessing the data. Many controls can work with `DataSet`s.

  ADO.NET offers a *strongly typed* `DataSet`, which is a special type of `DataSet`. A regular `DataSet` uses generic names for its contents — for example, Table1, column1, or column2. In a strongly typed `DataSet`, generic placeholder names are replaced with meaningful names from the data source, such as replacing Table1 with Customer table, and column1 or column2 with CustID or CompanyName.

  You can use the Visual Studio TableAdapter Configuration Wizard to work with tables in `DataSet`s. The wizard builds all the commands to retrieve and update the table's underlying data source.

- `DataReader`**:** Downloads your data but doesn't have an offline storage mechanism. When you use a `DataReader`, ADO.NET expects you to have a place to store the data in your application and a means of accessing the data from that store. Although a DataReader is much faster than a DataSet, a DataSet is more convenient.

The wizards and controls in Visual Studio work with DataSets. See Book V, Chapter 3 for information on working with DataSets and DataReaders in your code.

The Data Source Configuration Wizard creates and configures the TableAdapter for retrieving data and filling a DataSet for storing the data offline. The section "Dragging and Dropping Data," later in this chapter, walks you through using the Data Source Configuration Wizard.

Displaying data

The user interfaces for Windows applications and Web applications are built by using controls, such as labels, text boxes, and buttons. ADO.NET provides an entire set of controls just for displaying data.

Different controls are used for Web Forms and for Windows Forms, as listed in Table 1-1.

Table 1-1 Data Controls for Windows Forms and Web Forms

Project Type	*Control*	*Description*
Windows	`DataGridView`	Displays data in a table
	`BindingSource`	Binds data to a control
	`BindingNavigator`	Provides navigation to records
	`ReportViewer`	Displays a reporting services report
Web	`GridView`	Displays data in a table
	`DataList`	Displays data using a customizable format
	`DetailsView`	Displays a single row of data; often used with a `GridView`
Project Type	`Control`	Description
	`FormView`	Displays a single record in a form
	`Repeater`	Displays data in a customizable list
	`ReportViewer`	Displays a reporting services report

ADO.NET data controls are designed to display data from a data source. *Data binding* connects a control to a data source. Any control can be data-bound, including everyday controls, such as labels, text boxes, and buttons.

Many of the Web data controls are customizable. Controls, such as the `Repeater` control, don't have any display properties. You define how your data appears by defining a template for the control. The template defines the HTML that is used to display the data. See Book V, Chapter 2 for an example.

Web controls are rendered as HTML markup. More data-specific controls are provided for Web Forms than for Windows Forms because rendering data via HTML is more challenging. Both Windows Forms and Web Forms can use any control to display data.

Book V, Chapter 2 demonstrates some of the ADO.NET data controls available for Windows Forms and Web Forms applications. Chapter 2 also shows you how to filter and sort by using data controls and also how to bind data to regular controls, such as labels and text boxes.

Updating data

Most applications do more than just display data. Many times, users create new data or manipulate existing data. Updating and adding data is similar to viewing data: You have to store your changed data offline in your application by using either an ADO.NET `DataSet` or your own, custom data store; connect to your data source; and then execute commands against the data source that update it to reflect the changes made in the offline data store.

Visual Studio provides the TableAdapter Configuration Wizard for updating data. Book V, Chapter 3 shows the wizard in action.

Exploring the Visual Studio data toolbox

Working with data in your application requires many steps. Visual Studio provides a nice set of tools that "wrap around" ADO.NET and hide from you many of the complexities of how ADO.NET works. Table 1-2 lists the tools and what they help you do.

Table 1-2 Visual Studio Tools for Using ADO.NET

Visual Studio Tool	*ADO.NET Feature*	*Where You Can See the Tool in Action*
Data controls	Display data	Chapter 2
Data Source Configuration Wizard	Manage data sources Retrieve data Store data in `DataSets`	Next section, Chapter 2, and Chapter 3
Data Sources window	Manage data sources Retrieve data	Next section and Chapter 2
TableAdapter Configuration Wizard	Retrieve data Update data	Chapter 3

Visual Studio also includes these tools:

- **DataSet Designer:** Create and modify `DataSet`s. See Book V, Chapter 3 for an extensive walkthrough.
- **XML Designers:** Choose from several designers for working with XML data. See Book V, Chapter 4 for an extensive walkthrough.
- **Server Explorer and Data Explorer:** Use Server Explorer and Data Explorer to manage database objects. See Book V, Chapter 6 for more information about working with databases.

Understanding Databases

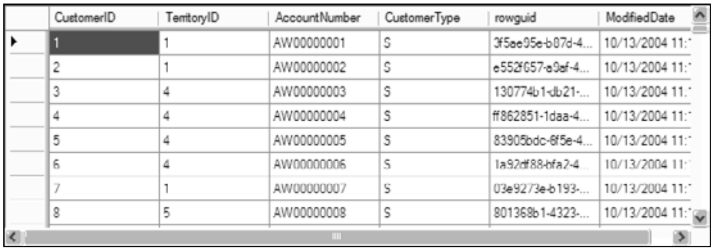
To be able to work with data, you have to know a little something about how it's stored in a relational database, especially if you want to use `DataSet`s. You often access databases, although Visual Studio can access more types of data sources than just databases. Even if data isn't stored in a database, you might decide to use a `DataSet` to store the data in your application's memory. A `DataSet` uses concepts that are similar to using a database.

A *database* is simply a place to store data. For example, you can store lists of information in a plain-text file, which is a database. Although plain-text files (often called *flat files*) might work fine for storing to-do lists, they don't work well for storing business data. That's why modern database management programs, such as Microsoft SQL Server, were created.

Introducing relational databases

Believe it or not, before relational database management systems, business data was stored in flat files, which are hard to work with. They aren't fast, and gathering data from multiple files is difficult, especially when no standard exists for determining how data is stored in the files.

Programmers, therefore, needed a means of not only storing data, but also accessing it again and associating it with other data in the database. The relational database management system was created to solve this problem. A *relational database management system,* or *RDBMS,* allows you to group common data records in a table. A table is made up of columns that represent fields in the records. Each record is a row in the table. For example, a database might have a Customer table and an Order table. Figure 1-1 shows an example of the Customer table.



CustomerID	TerritoryID	AccountNumber	CustomerType	rowguid	ModifiedDate
1	1	AW00000001	S	3f5ae95e-b87d-4...	10/13/2004 11:
2	1	AW00000002	S	e552f657-a9af-4...	10/13/2004 11:
3	4	AW00000003	S	130774b1-db21-...	10/13/2004 11:
4	4	AW00000004	S	ff862851-1daa-4...	10/13/2004 11:
5	4	AW00000005	S	83905bdc-6f5e-4...	10/13/2004 11:
6	4	AW00000006	S	1a92df88-bfa2-4...	10/13/2004 11:
7	1	AW00000007	S	03e9273e-b193-...	10/13/2004 11:
8	5	AW00000008	S	801368b1-4323-...	10/13/2004 11:

Figure 1-1: The Customer table comprises rows and columns.

People who work with data commonly show how a record in one table relates to another record in another table. For example, someone in sales may want to know how many customers placed orders during the preceding quarter. By using a relational database management system, you can flag columns in your table so that you can answer this kind of question.

The flagged columns are primary keys and foreign keys. When a column in a table is marked as a *primary key,* each row in the table must have a unique value in its primary key column. For example, in a Customer table, the CustomerID column must be unique if the column is marked as the table's primary key. You use the primary key to look up values in the Customer table and know that you can find but one unique entry.

You can reference a table's primary key in another table. For example, in the Order table, you need to know who the customer is for the order. Rather than store the customer's name, which can change, you can store the customer's `CustomerID` value so that you can look up the customer's information by using the Customer table. In the Order table, `Customer ID` is a *foreign key* because it's another table's primary key (the Customer table) referenced in a different table. Figure 1-2 shows an example of primary and foreign keys.

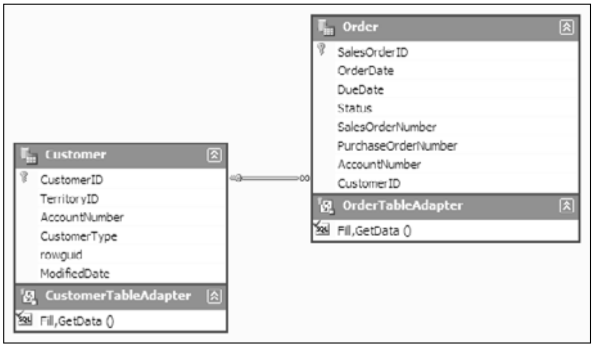


Figure 1-2: Use primary and foreign keys to create associations between tables.

Understanding SQL

RDBMS is a standardized set of commands for creating tables and columns and for inserting, updating, and reading data. This set of commands is *Structured Query Language (SQL)*.

TIP

You can pronounce SQL as either *es-cue-el* or *SEE-quel.* Typically, always spell the acronym (the former) when referring to the language. However, when talking about Microsoft SQL Server, use the latter. That way, you can differentiate between the standard and the product.

To see some examples of SQL, see Book V, Chapter 6. You might be amazed at the number of professional developers who shun SQL, preferring instead to use graphical tools to build their queries. Although graphical tools are easy to use, you need to understand the basics of SQL.

Relational database management systems consist of more than just tables. Most RDBMSs include these elements:

- **Administrative tools:** Manage database objects, such as tables, queries, and users.
- **Wizards:** Add database objects, back up databases, and schedule administrative services.
- **Visual tools:** Build queries and diagrams of the database structure.
- **Rules:** Protect your data from being erroneously deleted.

Some popular RDBMSs are Microsoft SQL Server 2008 (see Book V, Chapter 6), Oracle 10x, and Sybase SQL Anywhere.

Other kinds of database management systems exist, in addition to relational database management systems. *Object* database management systems, for example, store data as objects instead of as tables. Object databases are becoming especially popular because many programs are written using object-oriented techniques. You can find more information about object databases at `www.odbms.org`.

Dragging and Dropping Data

Visual Studio provides wizards you can use to add data to your Windows and Web applications. Adding data to an application is similar, whether it's a Windows Form or a Web Form. Essentially, adding data is as simple as dragging and dropping a control onto your form. After you drop the control, use the Data Source Configuration Wizard to

- Connect to a data source.
- Select tables or build a query from the data source.

- Store data in an offline data store.
- Bind data sources to a control to display the data.

Preparing your computer to work with data

The examples in this section use the Adventure Works sample database, along with Microsoft SQL Server 2008 Express Edition. You also need to add the current user when the Provisioning Tool runs.

This book uses the Adventure Works sample database, rather than the usual Northwind sample database, because Microsoft recommends using Adventure Works with SQL Server 2008. The Adventure Works database has been tuned to work with SQL Server 2008 and show off some of the features of SQL Server 2008.

If you didn't install SQL Server Express with Visual Studio 2010, see Book V, Chapter 6 for more information on installing SQL Server Express. The examples in this section also work with SQL Server 2008 Developer Edition.

To download the Adventure Works database, go to

```
http://codeplex.com/SqlServerSamples
```

Download and install the AdventureWorksDB.msi file. See Book V, Chapter 6 for more information about the SQL Server samples.

Adding data to Windows applications

To add data from the Adventure Works database to a Windows Form, follow these steps:

1. **Create a new Windows application.**

   See Book III, Chapter 1 for more information on creating Windows applications.

2. **Drag and drop a `DataGridView` control from the Data section of the toolbox to a blank Windows Form.**

   If the toolbox is closed, press Ctrl+Alt+X to open it.

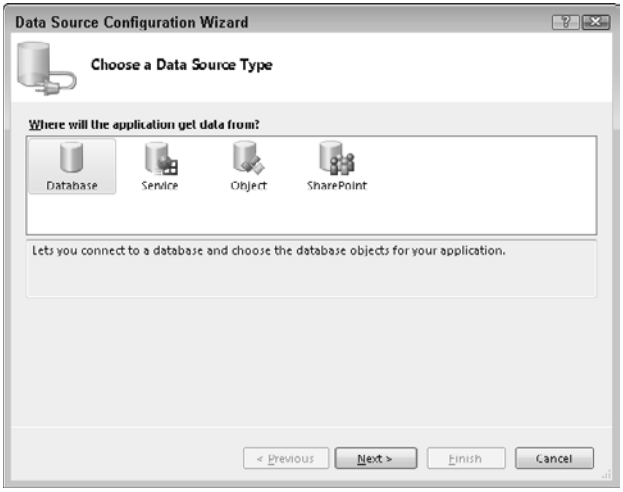
3. **Click the arrow in the upper-right corner of the `DataGridView` control to display the Common GridView Tasks dialog box.**

4. **Click the Choose Data Source drop-down list.**

   A list of the project's data sources appears. The list should be blank unless you already added data sources.

Before you can select data to display, you must configure the Data Source Configuration Wizard by following these steps:

1. **Click the Database icon on the Choose a Data Source Type page of the wizard (see Figure 1-3) and then click Next to go to the Choose Your Data Connection step.**

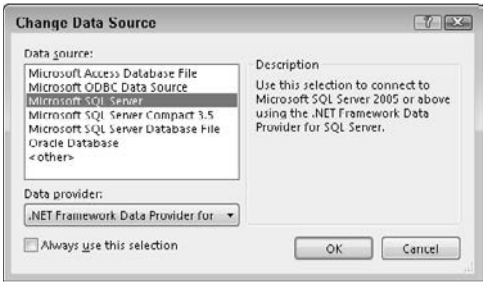

Data Source Configuration Wizard
Choose a Data Source Type
Where will the application get data from?
Database
Service
Object
SharePoint
< Previous
Next >
Finish
Cancel

Lets you connect to a database and choose the database objects for your application.

Figure 1-3: Select the Database object.

2. **Click the New Connection button to create a new database connection.**
3. **Make sure that the Data Source field is set to Microsoft SQL Server, as shown in Figure 1-4; otherwise, click the OK button and set the data source to Microsoft SQL Server.**

   The Add Connection dialog box appears.


Change Data Source
Data source:
Microsoft Access Database File
Microsoft ODBC Data Source
Microsoft SQL Server
Microsoft SQL Server Compact 3.5
Microsoft SQL Server Database File
Oracle Database
<other>
Description
Data provider:
.NET Framework Data Provider for
Always use this selection
OK
Cancel

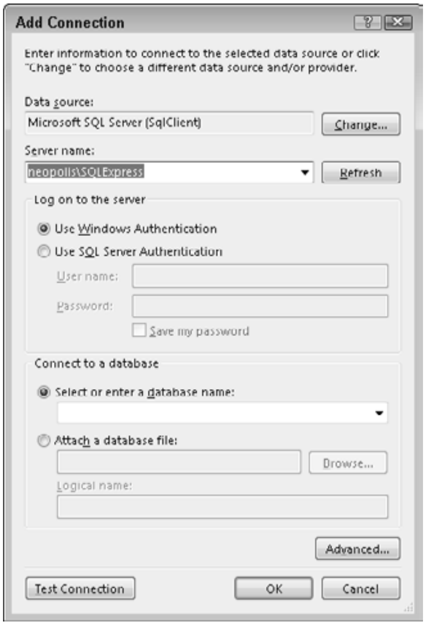
Use this selection to connect to Microsoft SQL Server 2005 or above using the .NET Framework Data Provider for SQL Server.

Figure 1-4: Make sure that the data source is set to Microsoft SQL Server.

4. **From the Server Name drop-down list, select your computer's name from the list.**

   You can also type **(local)** or a period for the server name to indicate that you're using the local server.

5. **Append** \SQLExpress **to the server name you entered in Step 4, as shown in Figure 1-5.**



Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft SQL Server (SqlClient)

Server name:
neopolis\SQLExpress

Log on to the server
● Use Windows Authentication
○ Use SQL Server Authentication
User name:
Password:
□ Save my password

Connect to a database
● Select or enter a database name:
○ Attach a database file:
Logical name:

Figure 1-5: Append the default name to the server name.

SQLExpress is the default name of the instance installed by SQL Server Express. (To read more about SQL Server instances, see Book V, Chapter 6.) Even though the Visual Studio documentation says that you don't need to enter the SQL Server instance, you do. Otherwise, Visual Studio can't connect to your database.

6. **Unless you have something other than a default installation, leave the Use Windows Authentication radio button selected (the default) in the Log On to the Server section.**

   See Book V, Chapter 6 to read more about Windows and SQL Server authentication.

7. **Select the Attach a Database File option in the Connect to a Database section and then click the Browse button to attach a database file.**

   The Select SQL Server Database File dialog box opens.

8. **Browse to the SQL Server database file directory where the Adventure Works sample is installed.**

   The default location for SQL Server databases is

   ```
   C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data
   ```

   TIP

   You may have to do a search because the path on my computer where it could be found was

   ```
   C:\Program Files (x86)\Microsoft SQL Server\90\Tools\Samples\
       AdventureWorks OLTP
   ```

9. **Select the `AdventureWorks_Data.mdf` database file and click Open.**

   The path to the Adventure Works database file appears in the Add Connection dialog box.

   If the file isn't on your computer, you can download it from

   ```
   www.microsoft.com/downloads/details.aspx?FamilyId=
       E719ECF7-9F46-4312-AF89-6AD8702E4E6E
   ```

10. **Click the Test Connection button.**

    If the connection is successful, you see the message `Test Connection Succeeded`. If the connection is unsuccessful, start troubleshooting your connection.

    See Book V, Chapter 6 for some tips on troubleshooting database connections.

11. **Click OK to close the test connection window.**

12. **Click OK to close the Add Connection dialog box.**

    The connection information appears in the Data Source Configuration Wizard, as shown in Figure 1-6.

13. **Click Next.**

    You're prompted to save the connection information.

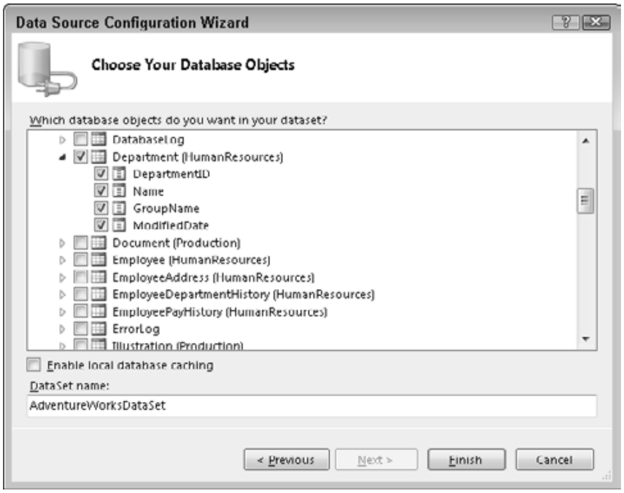
14. **Leave the Save the Connection option selected and then click Next.**

    The Choose Your Database Objects page appears.

15. **In the database objects pane, click the plus sign (+) next to Tables.**

    The list of database tables expands.

16. **Select the Department table and then click Finish to add the data to the form.**



Data Source Configuration Wizard

Choose Your Database Objects

Which database objects do you want in your dataset?

- □ DatabaseLog
- ☑ Department (HumanResources)
  - ☑ DepartmentID
  - ☑ Name
  - ☑ GroupName
  - ☑ ModifiedDate
- □ Document (Production)
- □ Employee (HumanResources)
- □ EmployeeAddress (HumanResources)
- □ EmployeeDepartmentHistory (HumanResources)
- □ EmployeePayHistory (HumanResources)
- □ ErrorLog
- □ Illustration (Production)

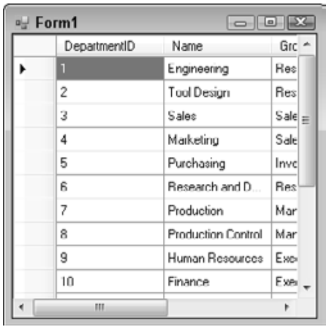
□ Enable local database caching

DataSet name: AdventureWorksDataSet

Figure 1-6: Select a table to add it to your form.

Visual Studio adds to your form the components that bind the `DataGridView` control to the AdventureWorks data source.

Run your form by pressing Ctrl+F5. Data from the Adventure Works Department table appears in the `DataGridView` control, as shown in Figure 1-7.



Form1

DepartmentID	Name	Grc
1	Engineering	Res
2	Tool Design	Res
3	Sales	Sale
4	Marketing	Sale
5	Purchasing	Invc
6	Research and D...	Res
7	Production	Mar
8	Production Control	Mar
9	Human Resources	Exe
10	Finance	Exe

Figure 1-7: Data from the Department table appears in the control.

To read more details about using the Data Source Configuration Wizard, adding data source connections, and working with other data-bound controls, see Book V, Chapter 2. The following section demonstrates how to use the Data Source Configuration Wizard to add data to a Web Form.

Adding data to Web Forms

You add data to Web Forms by using the Data Source Configuration Wizard. You can also use the wizard to add data to Windows Forms; however, selecting database objects, such as tables, varies slightly for Web Forms.

To add data to a Web Form using the Data Source Configuration Wizard, follow these steps:

1. **Create a new Web site.**

   See Book IV, Chapter 4 for more information on creating Web sites.

2. **Drag and drop a `GridView` control from the Data group in the Visual Studio toolbox to the Web Form.**

3. **Right-click the arrow in the upper-right corner of the `GridView` control to view the GridView Tasks dialog box (see Figure 1-8).**

4. **Select New Data Source from the Choose Data Source drop-down list.**

   The Data Source Configuration Wizard appears.

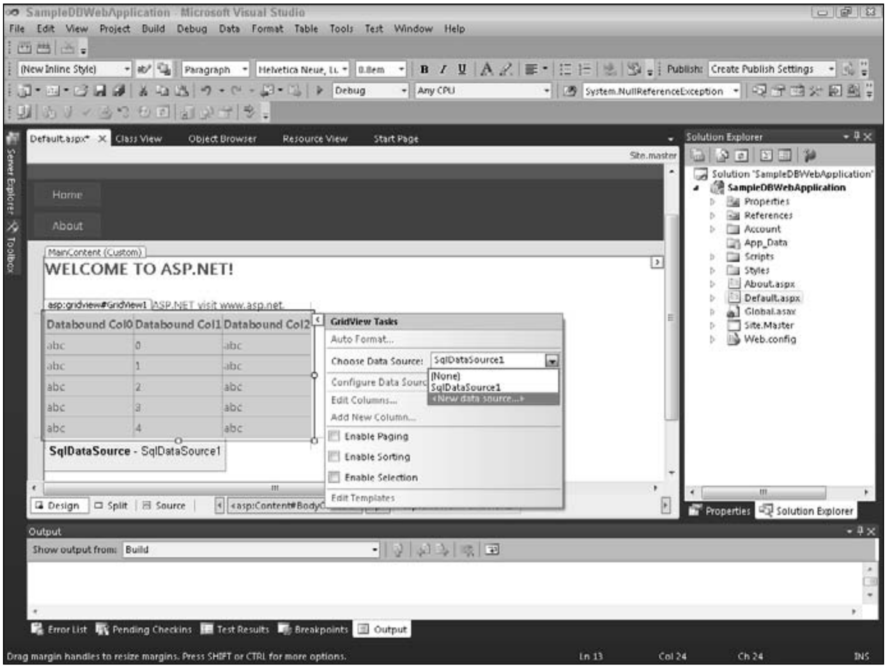


Figure 1-8: Add a new data source via the GridView Tasks dialog box.

5. **Click the Database icon for the data source and then click OK.**

6. **On the Choose Your Data Connection page, select your data connection.**

   If you completed the steps for the Windows section, click the drop-down arrow to reuse that connection. Otherwise, follow Steps 2 through 12 in the preceding section to create a new data connection to the Adventure Works database.

7. **Click Next.**

   You're prompted to save the connection string.

8. **Click Next again.**

   The next page of the wizard, Configure the Select Statement, appears.

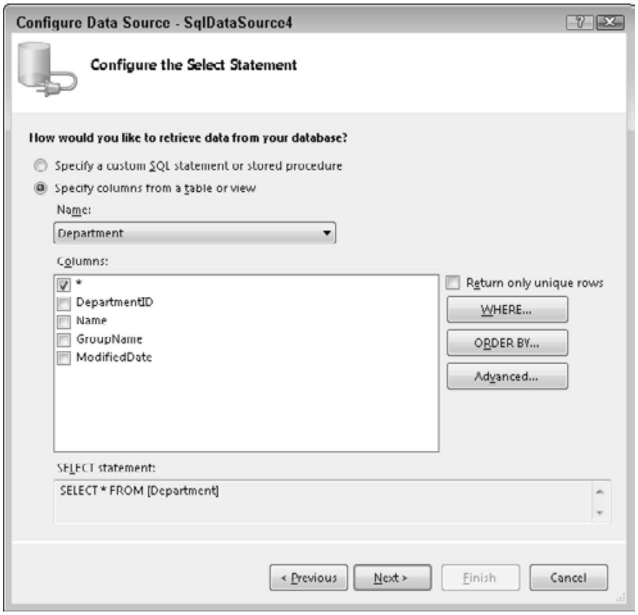
9. **Click the drop-down list of tables and choose Department.**

   The list of columns in the Department field appears.

10. **Select the asterisk check box to select all the columns, as shown in Figure 1-9.**

11. **Click Next.**

    The Test Query page appears.



Configure Data Source - SqlDataSource4

Configure the Select Statement

How would you like to retrieve data from your database?

○ Specify a custom SQL statement or stored procedure

● Specify columns from a table or view

Name: Department

Columns:

☑ *
☐ DepartmentID
☐ Name
☐ GroupName
☐ ModifiedDate

☐ Return only unique rows

SELECT statement:

SELECT * FROM [Department]

Figure 1-9: Select the columns you want to appear in the control.

12. **Click the Test Query button to test the query.**

    An error message might read that Department is an invalid object name. A bug in the Data Source Configuration Wizard makes it not work properly with databases that use schemas. To read more about configuring the Adventure Works database to work around the schemas, see Book V, Chapter 6. You can work around this error after completing the wizard.

13. **Click Finish.**

    The Data Source Configuration Wizard closes.

Visual Studio adds the following declarative syntax to the Web Form to make it display the data in the GridView control:

```
<asp:GridView ID="GridView1" runat="server" DataSourceID="SqlDataSource1">
</asp:GridView>
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString=
     "<%$ ConnectionStrings:AdventureWorksConnectionString %>"
    SelectCommand="SELECT * FROM [Department]">
</asp:SqlDataSource>
```

For the GridView control to access the Department table, the SelectCommand property must be more specific about how to reach the table. To make the SelectCommand property more specific, change it to

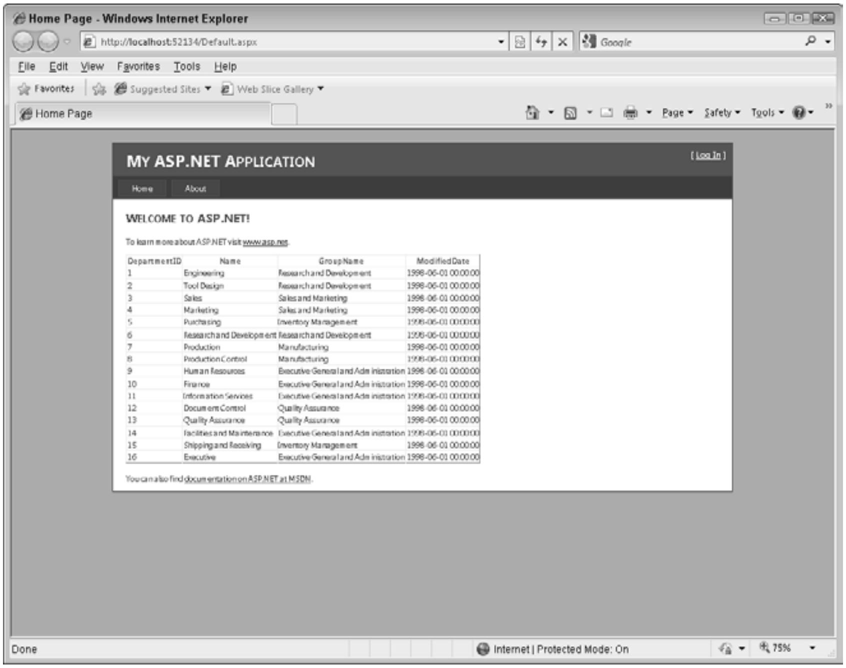
```
SelectCommand="SELECT * FROM [HumanResources].[Department]"
```

The HumanResources schema is associated with the Department table in the Adventure Works database. [HumanResources].[Department] is the *fully qualified* name because it makes it clear how the table is accessed. Whenever you have trouble accessing a database table, make sure that you can access it by using its fully qualified name.

If you receive invalid-object errors while working with the Data Source Configuration Wizard in ASP.NET, make sure that the object's name is fully qualified and spelled correctly (including the use of upper- and lowercase).

Alternatively, you can use the Query Builder to build the select statement in Step 10 in the preceding set of steps. The Query Builder correctly builds the select statement.

Press Ctrl+F5 to run the Web Form. The data from the Department table appears, as shown in Figure 1-10.



WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

DepartmentID	Name	GroupName	ModifiedDate
1	Engineering	Research and Development	1998-06-01 00:00:00
2	Tool Design	Research and Development	1998-06-01 00:00:00
3	Sales	Sales and Marketing	1998-06-01 00:00:00
4	Marketing	Sales and Marketing	1998-06-01 00:00:00
5	Purchasing	Inventory Management	1998-06-01 00:00:00
6	Research and Development	Research and Development	1998-06-01 00:00:00
7	Production	Manufacturing	1998-06-01 00:00:00
8	Production Control	Manufacturing	1998-06-01 00:00:00
9	Human Resources	Executive General and Administration	1998-06-01 00:00:00
10	Finance	Executive General and Administration	1998-06-01 00:00:00
11	Information Services	Executive General and Administration	1998-06-01 00:00:00
12	Document Control	Quality Assurance	1998-06-01 00:00:00
13	Quality Assurance	Quality Assurance	1998-06-01 00:00:00
14	Facilities and Maintenance	Executive General and Administration	1998-06-01 00:00:00
15	Shipping and Receiving	Inventory Management	1998-06-01 00:00:00
16	Executive	Executive General and Administration	1998-06-01 00:00:00

You can also find documentation on ASP.NET at MSDN.

Figure 1-10: The Department data appears in the Web Form's `GridView` control.

Chapter 2: Show Me the Data

In This Chapter

- **Accessing data in a Windows application**
- **Taking a closer look at data binding**
- **Creating a data-centric Web site**

Data is data is data, or so you would think. When you're working with data in Visual Studio, the approach you take for Windows Forms varies slightly from the one for Web applications. Although the Windows Forms Designer sports the Data Sources window, Visual Web Developer has no such jumping-off point for managing all your project's data sources.

Both designers share some tools, such as the Data Source Configuration Wizard, but the experience in Visual Web Developer is less refined. The data tools in Windows Forms Designer integrate very well and flow seamlessly. Rest assured, whether you're building Windows applications or Web sites, Visual Studio has a data tool to get the job done.

This chapter walks you through the procedure for working with the set of data tools in Visual Studio. Don't be surprised to see the same tools demonstrated once for Windows applications and again for Web sites. Despite having the same names, sometimes the tools behave completely differently.

Adding Data Access in Windows Applications

Visual Studio provides plenty of ways to add access to data in a Windows application. Here are the primary ways to do so by using the Windows Forms Designer:

- **Drag and drop a data control onto a form.**

  You can read how to access data by using the `GridView` control in Book V, Chapter 1.
- **Use the Data Source Configuration Wizard.**
- **Use the Data Sources pane.**

Data sources are any source of data you want to access in your application. Examples of data sources include databases, Web services, objects, and XML files. You can use a Visual Studio wizard to access almost every form of data you can imagine.

TIP

You aren't restricted to just accessing Microsoft's data offerings, such as SQL Server and Access. You can connect to any vendor's database management system by creating a data connection, as described in the upcoming section "Connecting to databases."

All the Visual Studio data tools are intertwined in such a way that even if you start out using one tool, you often end up finishing the task by using another wizard altogether.

Regardless of which tool you use, you always start by specifying a data source. After configuring the data source, you configure a control to use with the data source. Visual Studio generates the code to take care of everything that happens between those two points.

Of course, you can always write your own code to access data. (Sigh.) If you insist on doing things the hard way, see Book V, Chapter 5 for examples of using the Code Editor to access data with ADO.NET. Sometimes, though, writing your own data access code is necessary to do things outside "normal data access."

The sections that follow focus on the maneuvers that Visual Studio makes so easy. Find out the basics of the Data Sources pane and how you use it to view, add, and connect to data sources.

Working with the Data Sources pane

The Data Sources pane is a handy tool for managing all your project's data sources. The Data Sources pane works hand in hand with the Data Source Configuration Wizard to create new data sources and configure access to the data.

You can use the Data Sources pane to work with data source objects. Open the window in one of two ways:

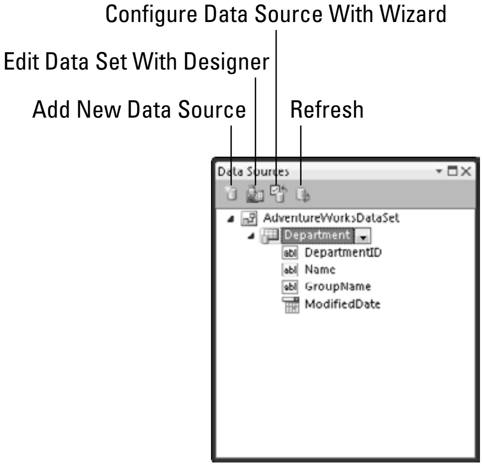
- Choose Data➪Show Data Sources.
- Press Shift+Alt+D.

Use the Data Sources pane to add new data sources to your project and work with existing data sources. The Data Sources toolbar, shown in Figure 2-1, has these commands, from left to right:

- **Add New Data Source:** Start the Data Source Configuration Wizard.
- **Edit DataSet with Designer:** Open the DataSet Designer with the selected `DataSet` from the Data Sources tree. See Chapter 3 of this book for more information about using the DataSet Designer.

- **Configure Data Source with Wizard:** Start the Data Source Configuration Wizard so that you can select database objects to add to the `DataSet`.
- **Refresh:** Refresh the data sources listed in the Data Sources tree.

Datasets are used to store data retrieved from data sources.


Configure Data Source With Wizard
Edit Data Set With Designer
Add New Data Source
Refresh

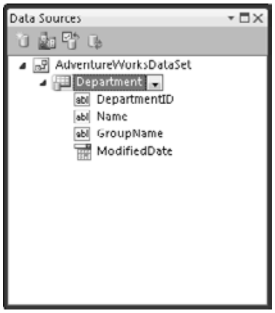
Data Sources

- AdventureWorksDataSet
  - Department
    - DepartmentID
    - Name
    - GroupName
    - ModifiedDate

Figure 2-1: Use the Data Sources toolbar to work with data sources.

Viewing data source objects

Data sources are listed in tree view in the Data Sources pane, shown in Figure 2-2.



Data Sources

- AdventureWorksDataSet
  - Department
    - DepartmentID
    - Name
    - GroupName
    - ModifiedDate

Figure 2-2: The Data Sources pane displays a project's data sources in tree view.

The Data Sources tree is a hierarchical view that shows these elements:

- **Expandable parent data sources:** A data source that uses `DataSet`s, such as a database, shows the `DataSet` as the parent. Objects and Web services show a namespace.
- **Expandable child objects nested within the parent:** A `DataSet` expands to list the tables contained with it.
- **A set of child objects:** A table within a `DataSet` expands to list the table's columns.

Adding data sources

The Data Sources pane uses the Data Source Configuration Wizard to add new data sources. To start the Data Source Configuration Wizard from the Data Sources pane, follow these steps:

1. **Create a new Windows project.**

   See Book III, Chapter 1 for more information on creating Windows projects.

2. **Choose Data➪Show Data Sources.**

   The Data Sources pane appears.

3. **Click the Add New Data Source button on the Data Sources toolbar.**

   The Data Source Configuration Wizard appears.

   Alternatively, you can choose Data➪Add New Data Source to start the Data Source Configuration Wizard. Data sources created with the wizard appear in the Data Sources pane.

To add a new data source by using the Data Source Configuration Wizard, follow these steps:

1. **On the Choose a Data Source Type page in the wizard, select a data source, as shown in Figure 2-3.**

   Your data source choices in the wizard are Database, Service, and Object.

2. **Click Next.**

   Depending on the data source you select in Step 1, the wizard displays one of the following pages to create your connection:

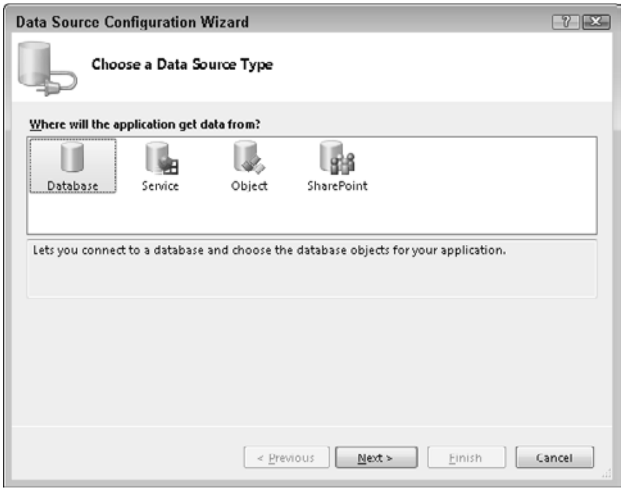
- *Database*: The Choose Your Data Connection page appears. Select an existing database connection or select New Connection to add a new database connection. See the following section for more about managing database connections.
- *Service*: Opens the Add Web Reference dialog box. Enter the URL for the Web service or browse for one. See Book III, Chapter 6 for more information about Web services.
- *Object*: The Select the Object You Wish to Bind To page appears. It displays a tree view of the classes in your project. Click the Add Reference button to add a reference to another project.
- *SharePoint:* Allows you to connect to a SharePoint site and choose SharePoint objects for your application.

After you establish a connection to the data source, the wizard displays a list of the data source's objects.

TIP

The generic term *object* refers to the contents of the data source. For example, the objects in a database are tables.

3. **Select from the data source the objects you want to use.**



Data Source Configuration Wizard

Choose a Data Source Type

Where will the application get data from?

Lets you connect to a database and choose the database objects for your application.

Figure 2-3: Select the data source in the wizard.

4. **Click Finish.**

The data source is added to the Data Sources pane. Figure 2-4 shows a Data Sources pane with a database and objects added to it.

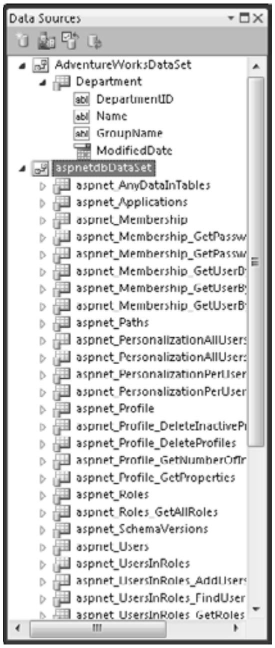

Data Sources
AdventureWorksDataSet
Department
DepartmentID
Name
GroupName
ModifiedDate
aspnetdbDataSet
aspnet_AnyDataInTables
aspnet_Applications
aspnet_Membership
aspnet_Membership_GetPassw
aspnet_Membership_GetPassw
aspnet_Membership_GetUserD
aspnet_Membership_GetUserB
aspnet_Membership_GetUserB
aspnet_Paths
aspnet_PersonalizationAllUsers
aspnet_PersonalizationAllUsers
aspnet_PersonalizationPerUser
aspnet_PersonalizationPerUser
aspnet_Profile
aspnet_Profile_DeleteInactivePr
aspnet_Profile_DeleteProfiles
aspnet_Profile_GetNumberOfIn
aspnet_Profile_GetProperties
aspnet_Roles
aspnet_Roles_GetAllRoles
aspnet_SchemaVersions
aspnet_Users
aspnet_UsersInRoles
aspnet_UsersInRoles_AddUsers
aspnet_UsersInRoles_FindUser
aspnet_UsersInRoles_GetRoles

Figure 2-4: The wizard adds data source objects to the Data Sources pane.

Connecting to databases

A database is a kind of data source — probably the most popular. Because databases are such popular data sources, Visual Studio provides a special dialog box for creating connections to databases.

You can access the Add Connections dialog box in several ways, including the ones listed here:

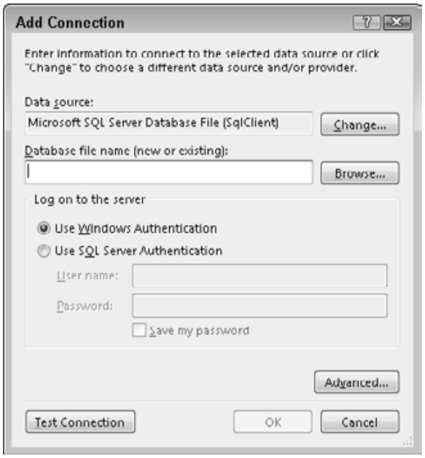
- Choose Tools➪Connect to Database.
- Use the Choose Your Data Connection page in the Data Source Configuration Wizard.
- Click the Connect to Database button on the Server Explorer toolbar.

Regardless of how you access the Add Connections dialog box, use it to make connections to all your project's databases.

TIP

Database connections are reusable. You don't have to create a new database connection each time you use the database as a data source.

Figure 2-5 shows the Add Connection dialog box for accessing a Microsoft SQL Server database. The choices available in the dialog box change to suit the selected database source.



Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server Database File (SqlClient)

Database file name (new or existing):

● Use Windows Authentication

○ Use SQL Server Authentication

□ Save my password

Figure 2-5: Use the Add Connection dialog box to create database connections.

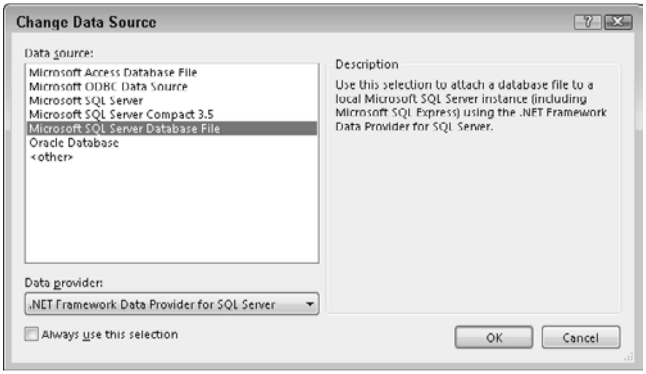
To change the database source in the Add Connection dialog box, follow these steps:

1. **Choose Tools➪Connect to Database.**

   The Add Connection dialog box appears.

2. **Click the Change button to change the data source.**

   The Change Data Source dialog box appears, as shown in Figure 2-6.



Data source:

Microsoft Access Database File
Microsoft ODBC Data Source
Microsoft SQL Server
Microsoft SQL Server Compact 3.5
Microsoft SQL Server Database File
Oracle Database
<other>

Description

Use this selection to attach a database file to a local Microsoft SQL Server instance (including Microsoft SQL Express) using the .NET Framework Data Provider for SQL Server.

Data provider: .NET Framework Data Provider for SQL Server

□ Always use this selection

Figure 2-6: Select a new data source from the Change Data Source dialog box.

The available data sources are

- Microsoft Access Database File
- Microsoft ODBC Data Source
- Microsoft SQL Server

- Microsoft SQL Server Compact 3.5
- Microsoft SQL Server Database File
- Oracle Database
- Other

TIP

You should pick the data source from the list that most closely matches your data source. For example, you should choose Microsoft SQL Server to connect to an SQL Server database rather than the more generic ODBC Data Source. Although you can use the Microsoft ODBC Data Source to access almost any data source imaginable, the other choices offer better performance with a specific kind of data source. Click a data source in the dialog box to view a description.

3. **Choose a data source from the list.**

   When you click a data source, the Change Data Source dialog box displays the data source's default data provider and description.

   A *data provider* is a component in *ADO.NET,* which is the data access technology in the Microsoft.NET framework. ADO.NET provides data providers for connecting to a database and executing commands against the database. The Add Connection dialog box configures the ADO.NET default data provider to work with the data source you select. For more information about data providers, see Chapter 5 in this book.

   TIP

   If needed, open the Data Provider drop-down list to select a different ADO.NET data provider to use with your data source. The data source's default data provider is selected automatically.

4. **Click OK to close the Change Data Source dialog box.**

   The Add Connection dialog box is updated to reflect the connection information required to connect to the data source.

5. **Use the Add Connection dialog box to finish configuring the connection.**

   The information required from the Add Connection dialog box varies, depending on the data source selected. All the data sources have these features:

   - *Advanced button:* View and modify the data source's connection information in a property grid.
   - *Login information***:** Specify the kind of login you want to use, as well as a username and password to use (if required).

6. **When you're done, click the Test Connection button in the Add Connection dialog box to make sure that the connection works without closing the dialog box.**

   TIP

   Any time you're having difficulties accessing a database, use the Add Connections dialog box to test your database connection. Without a working database connection, you can't access the database.

ABCs of ODBC

ODBC — *Open Database Connectivity* — is a standard that has been used for many years to access everything from Access databases to text files. ODBC is a good choice when you need to access a data source for which Visual Studio doesn't already provide a way to access.

Use the Add Connections dialog box to create your connections, even if you plan to write code to access data. You can use the Add Connections dialog box to test your connection, which saves you from having to do a lot of troubleshooting if your code isn't working properly.

The Add Connections dialog box creates a connection string that specifies how your application should access the data source. *Connection strings* store everything you type in the Add Connections dialog box, including server names, usernames, and passwords. You have to manage connection strings securely. To read more about managing connection strings, see Book V, Chapter 5.

Adding controls from the Data Sources pane

The Data Sources pane is much more than a container for storing your project's data sources. You can drag and drop data from the Data Sources pane onto the Windows Forms Designer. Visual Studio adds a databound control to your form and "wires" it to your data source so that it magically works.

A control is *databound* if it retrieves data from a data source. A control's properties can also be databound. See the section "Binding by using the Binding property," later in this chapter, to see how to bind data to a control's property.

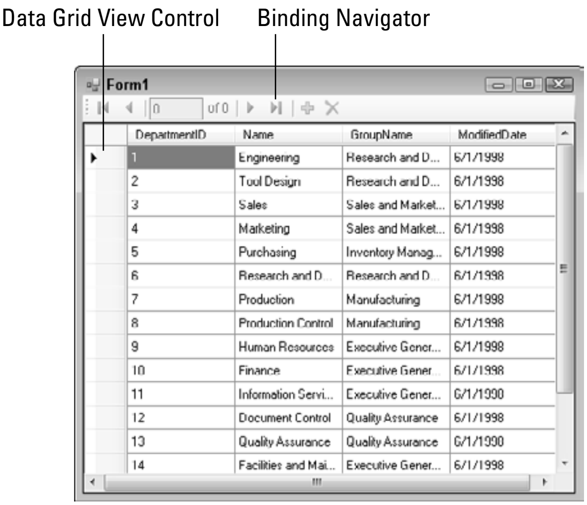
To add a databound control from the Data Sources pane, follow these steps:

1. **Use the Data Source Configuration Wizard to add the Department table from the Adventure Works database by following the steps described in the "Adding data sources" section, earlier in this chapter.**

   This example uses the Adventure Works sample database from Microsoft. See Book V, Chapter 1 for information about downloading and installing the sample.

2. **Choose Data➪Show Data Sources to display the Data Sources pane.**
3. **In the Data Sources pane, expand the AdventureWorks data source by clicking the plus sign (+) next to the data source.**
4. **Drag and drop the Department table on the Windows Form.**

Visual Studio adds a `DataGridView` control and a `BindingNavigator` component to the form, as shown in Figure 2-7.



DepartmentID	Name	GroupName	ModifiedDate
1	Engineering	Research and D...	6/1/1998
2	Tool Design	Research and D...	6/1/1998
3	Sales	Sales and Market...	6/1/1998
4	Marketing	Sales and Market...	6/1/1998
5	Purchasing	Inventory Manag...	6/1/1998
6	Research and D...	Research and D...	6/1/1998
7	Production	Manufacturing	6/1/1998
8	Production Control	Manufacturing	6/1/1998
9	Human Resources	Executive Gener...	6/1/1998
10	Finance	Executive Gener...	6/1/1998
11	Information Servi...	Executive Gener...	6/1/1998
12	Document Control	Quality Assurance	6/1/1998
13	Quality Assurance	Quality Assurance	6/1/1998
14	Facilities and Mai...	Executive Gener...	6/1/1998

Figure 2-7: Drag and drop data objects from the Data Sources pane to create databound controls.

Alternatively, you can expand the Department table in the Data Sources pane and add columns individually to the form.

5. **Press Ctrl+F5 to run the form.**

You will see the data populating the control. In addition to the `DataGridView` control, the Data Sources pane automatically adds several components to the Windows Form:

- `BindingSource`: Connects data sources to databound controls.
- `BindingNavigator`: Provides built-in record navigation, including buttons to add and delete records.
- `TableAdapter`: Builds the commands for retrieving and updating data from the data source.

These components are responsible for wiring your data source to your controls so that they "just work." You don't have to write any code to retrieve your data or update it. Simply drag and drop the data objects from the Data Sources pane onto the Windows Forms Designer. Visual Studio takes care of all the details, and you get all the credit!

TIP

You can view these components in the lower part of the designer screen.

Visual Studio adds a `DataGridView` control by default when you drop data from the Data Sources pane. You can specify the control you want to add to the Windows Form. To set controls for items in the Data Sources pane, follow these steps:

1. **Click the drop-down arrow for the data source in the Data Sources pane.**

   A list of available controls appears.

   You can work with a parent container, such as a database table, or you can set controls for individual columns in the table.

2. **Select a control from the drop-down list, as shown in Figure 2-8.**

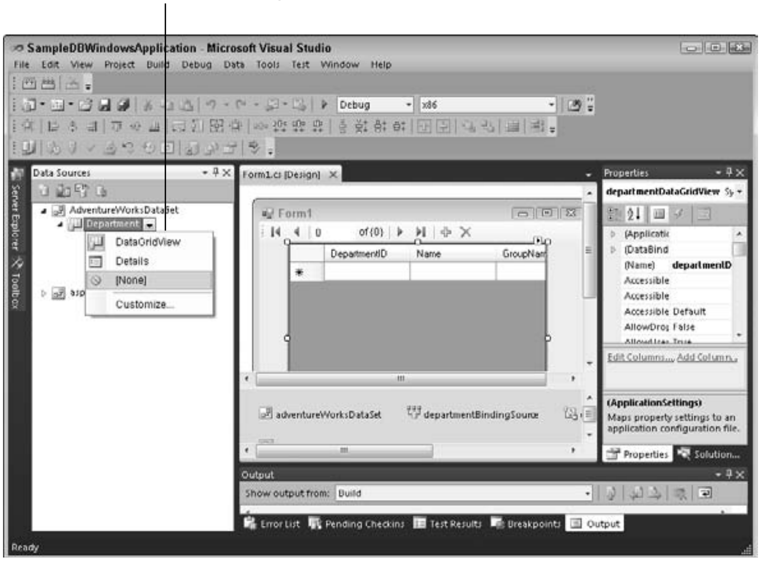

Select a control from the drop-down list

Figure 2-8: Click the drop-down arrow to change the default control that's added to the Windows Form.

   When you drag the data object onto the form, the selected control appears.

3. **Set the parent container's control to Details to drop the parent's child controls on the form.**

The kinds of controls that are listed depend on the type of data in the data source. For example, a column that stores date-and-time data lists the `DateTimePicker` control. To use the shortcut menu to select a different control, right-click the data item in the Data Sources pane and then either

- **Choose None** to display no control for the data.
- **Choose Customize** to add a control to the list.

Getting Your Data in a Bind

Binding is the process of "wiring" a data source to a data control. Visual Studio provides several tools that make quick work of binding data sources to data controls, thus making it possible to perform the following actions:

- Navigate
- Update
- Delete
- Sort
- Filter

ADO.NET provides two components dedicated to providing binding services that make databound controls come alive with little or no coding:

- `BindingSource`: Links the data source and the databound control to make it possible to access and manipulate data
- `BindingNavigator`: Provides navigation to the data source linked to the `BindingSource` component

Visual Studio creates the `BindingSource` and `BindingNavigator` components any time you drag and drop data from the Data Sources pane.

Using `BindingSource` is the recommended method for binding data in Windows applications, regardless of the data source. No similar component exists for Web applications.

Using BindingSource to connect to other data

The `BindingSource` component, which was new to Visual Studio 2008, sits between your data source and the databound controls. The `BindingSource` component wires your data source to a control that displays the data. You can use `BindingSource` to

- Bind data to your controls without writing code.
- Change the data source without breaking the databound controls.
- Sort, filter, and add new data.

The `BindingSource` component simplifies working with data by exposing properties for common data-manipulation tasks, such as sorting and filtering data.

`BindingSource` is called a component instead of a control because no onscreen display is associated with it. A control is a specific kind of component — one that you use to build your user interface.

The easiest way to work with the `BindingSource` component is to drag and drop data from the Data Sources window. The following examples show you how to use `BindingSource` to sort and filter properties:

1. **Display the properties of a `BindingSource` component by selecting the component in the designer and then pressing F4.**

   The Properties window appears.

   For example, to use the `BindingSource` component, you could create a `departmentBindingSource` in the earlier section "Adding controls from the Data Sources pane," click the component, and then press F4.

2. **In the `Filter` property, type a value to use to filter the data source.**

   For example, to filter the GroupName column in the `DepartmentBindingSource`, enter **a** GroupName like

   ```
   Executive%
   ```

   The *like operator* (such as =) checks whether a value matches a pattern.

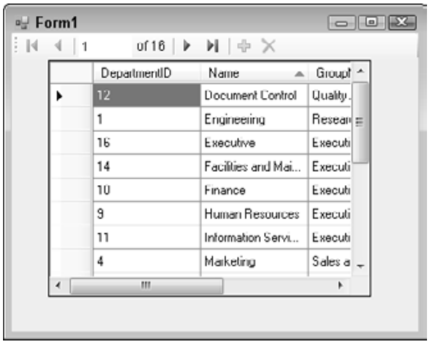
3. **In the `Sort` property, type the name of a column to sort by.**

   For example, enter **Name asc** to sort the Name column in ascending order.

   `asc` is short for ascending, and `desc` is short for descending.

4. **Press Ctrl+F5 to run your form.**

   The list of departments is filtered on the GroupName column and sorted in ascending order by name, as shown in Figure 2-9.



DepartmentID	Name	GroupN
12	Document Control	Quality ..
1	Engineering	Resear
16	Executive	Executi
14	Facilities and Mai...	Executi
10	Finance	Executi
9	Human Resources	Executi
11	Information Servi...	Executi
4	Marketing	Sales a

Figure 2-9: The data is filtered and sorted based on properties set in the `Binding-Source` component.

Two other important properties for the BindingSource component are

- DataSource: Specifies the data source used by BindingSource
- DataMember: Specifies which table or object within the DataSource to use to get the data

In the preceding example, DataSource is set to AdventureWorksDataSet. The DataMember property is Department. You can add another table to the DataSet and specify the DataMember property for the table.

You can use BindingSource features to step forward and backward through a set of records and add, update, and delete data. When a BindingNavigator component is used with BindingSource, its data-navigation and -manipulation features are automatically mapped to the BindingNavigator toolbar buttons.

Using BindingNavigator

The BindingNavigator component is a built-in menu control for navigating data. You can find the BindingNavigator control in the Data tab of the Visual Studio toolbox. The control, as shown in Figure 2-10, includes the following buttons (from left to right) for navigating and manipulating data:

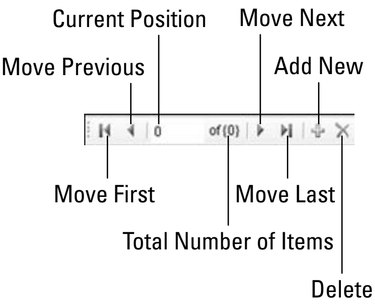


Figure 2-10: Use this control to navigate and manipulate data.

- Move First
- Move Previous
- Current Position
- Total Number of Items
- Move Next
- Move Last

- Add New
- Delete

The BindingNavigator component is used in conjunction with BindingSource. The component has a BindingSource property that, when set to a BindingSource component, allows you to navigate and manipulate data.

Clicking a button on BindingNavigator triggers events in BindingSource. For example, when a user clicks the Add New button on BindingNavigator, the AddNew method in BindingSource is called. You don't have to worry about making it work. Just setting the BindingSource property for BindingNavigator wires the two components together. Better yet, just dragging and dropping objects from the Data Sources pane makes Visual Studio create and configure these two components for you.

TIP

To prevent users from adding data to the underlying data source, set the AllowNew property of BindingSource to False. The Add New button is automatically disabled in BindingNavigator.

Binding by using the Binding property

Any control's property can be bound to data. You don't always have to use special data controls to work with data. In the example in this section, you can see how to bind a department's name to a label control's Text property:

1. **Drag and drop a label control onto a Windows Form.**
2. **Open the label control's properties by pressing F4.**

   The Properties window appears.
3. **Click the plus sign (+) to expand the `DataBindings` property.**
4. **Click the down arrow for the `Text` property in the DataBindings section.**

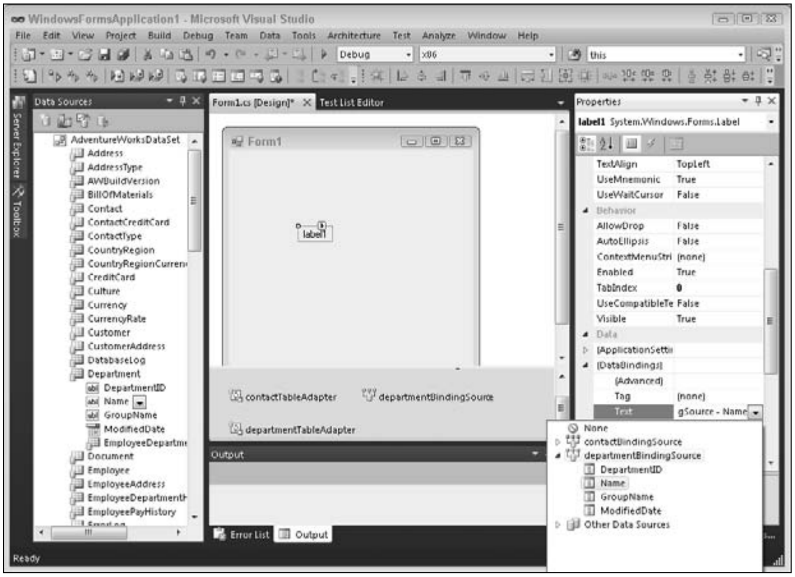
   A list of data sources appears.

   TIP

   To access additional properties you can create data binding for, click the ellipsis button for the Advanced property.
5. **Select a data source from the drop-down list or click the Add Project Data Source link to add a new data source.**

   If you add a new data source, the Data Source Configuration Wizard appears. You step through the wizard and add your data.
6. **Expand the data source and select the column you wish to bind the property to.**

For example, select the Name column in the Department table to bind the `Text` property to it, as shown in Figure 2-11.



TextAlign	TopLeft
UseMnemonic	True
UseWaitCursor	False
Behavior	
AllowDrop	False
AutoEllipsis	False
ContextMenuStri	(none)
Enabled	True
TabIndex	0
UseCompatibleTe	False
Visible	True
Data	
(ApplicationSetti	
(DataBindings)	
(Advanced)	
Tag	(none)
Text	gSource - Name

Figure 2-11: Select the Name column from the data source.

The text `DepartmentBindingSource — Name` is added to the property's value.

7. **Press Ctrl+F5 to run the form.**

   The value of the first record in the data source appears in the label control's `Text` property.

Using a `BindingSource` component as the data source in Step 5 enables you to step through the records on your control with a `BindingNavigator`. By using the `BindingSource` as your data source for multiple controls, you can synchronize the data displayed by the controls. For example, if you add a `DataGridView` control to your form, you could use a `DepartmentBindingSource` from the Data Sources pane. Then the `DataGridView` and the label control both display the same record. As you navigate through the records, subsequent records are shown after clicking the Navigate button, as shown in Figure 2-12.

Data binding to properties is useful any time you want to dynamically populate properties.

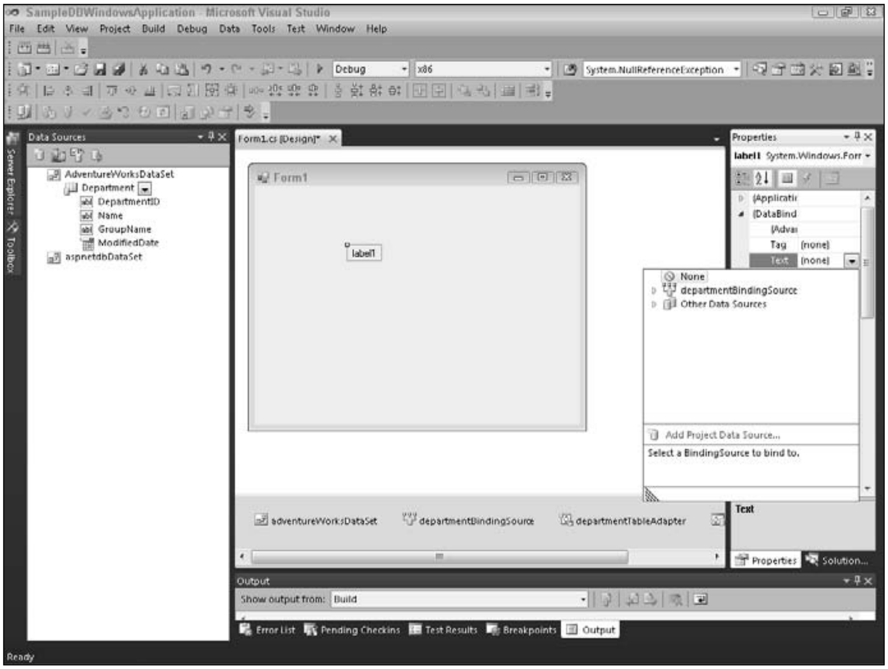


Figure 2-12: Both controls display data for the current record.

Creating Web Forms That Access Data

Developing Web applications is challenging because you have to send information across a network. Of course, this isn't just any network. Your Web pages are sent out by way of the world's network, the Internet.

Processing a Web page involves these stages:

1. A client's browser sends a request to a server for the Web page.
2. The server responds to the client and sends the contents of the Web page.
3. The client's browser receives the Web page and displays its content.

Repeat this process 100 bazillion times per day, and you have some idea of the amount of requests and responses bouncing around on the Internet.

The requests and responses of clients and servers are controlled by a set of rules named *HyperText Transfer Protocol* (HTTP). One characteristic of HTTP is that it's *stateless,* which means that after a request or response is processed, HTTP forgets that the response or request ever existed. If clients and servers on the Web suffer from amnesia with every click of the mouse, how is it possible to get data from servers to clients and back again?

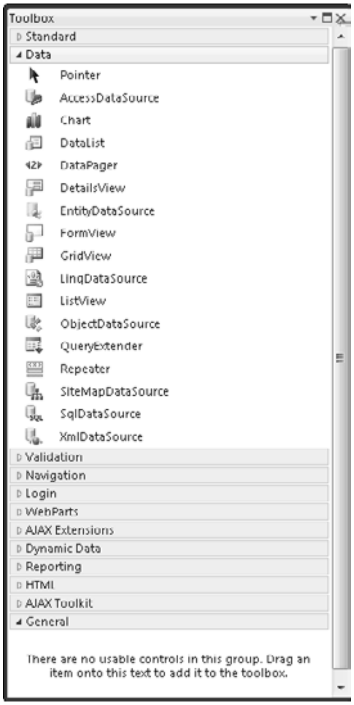
When you want to make data work on the Web, three players do the heavy-duty lifting of making data-centric Web applications work:

- ASP.NET provides features that overcome HTTP's inability to maintain state.
- ADO.NET provides lots of widgets and commands for getting data from data sources, such as databases and XML files.
- Visual Studio provides the designer tools that make light work of using the high-tech gadgetry that makes Web applications work.

In this section, you can read how to use Visual Studio to access data on your Web pages. You can sleep better at night knowing that Visual Studio is protecting you from dealing with ASP.NET, ADO.NET, and HTTP; or, you can have your own case of amnesia and happily get on with building data-centric Web applications with Visual Studio.

Meet the controls

The Visual Studio toolbox includes many data controls for displaying data and creating data sources. All the data controls are grouped on the toolbox's Data tab, as shown in Figure 2-13. To open the toolbox, choose View⇨Toolbox or press Ctrl+Alt+X.



There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

Figure 2-13: Data controls are grouped on the Data tab of this Visual Studio toolbox.

Five controls display data on a Web page. They're often called *databound* controls because data is bound to them:

- `GridView`: Displays a tabular view of data.
- `DataList`: Displays data in a list with a customizable format.
- `DetailsView`: Displays a single record at a time in a table row. This control is often used with `GridView` to create master/detail views.
- `FormView`: Displays a single record without specifying how the data is displayed. The display can be customized.
- `Repeater`: Displays data in a list that can be customized.

Each of these controls uses HTML markup to render your data to the client's Web browser. By default, the text is rendered by using table markup. All these controls can be customized, so you can specify the exact HTML markup that you want ASP.NET to use when it sends the page to the browser.

Visual Studio provides a data source control for each kind of data source you can access in ASP.NET. The five controls for creating data sources are

- `SqlDataSource`: Accesses a database, such as Microsoft SQL Server or Oracle
- `AccessDataSource`: Accesses a Microsoft Access database
- `LinqDataSource`: Accesses data from a LINQ query
- `ObjectDataSource`: Accesses data in a business object
- `XmlDataSource`: Accesses data in an XML file
- `SiteMapDataSource`: Creates a data source to access a site map

TECHNICAL STUFF

The data source controls are comparable to the `BindingSource` component used with Windows Forms. The obvious difference, of course, is that `BindingSource` works with multiple data sources, whereas the ASP.NET data source controls are specialized to work with specific data sources.

Going to the source

You use data source controls to configure access to a particular type of data source. Each control, except for the `SiteMapDataSource` control, has a configuration wizard that you use to configure the data source control. The `SiteMapDataSource` control doesn't need a wizard because it magically works with the site's site map. See Book III, Chapter 6 for more about site maps.

To launch the configuration wizard for any of the data source controls, follow these steps:

1. **Drag and drop one of the data source controls onto a Web Form.**
2. **Click the arrow in the upper-right corner of the control to display a context menu.**
3. **Click the Configure Data Source link in the context menu, as shown in Figure 2-14.**
4. **Step through the wizard to configure the data source.**

Data source controls often store their results in `DataSet`s. By using `DataSet`s, you can easily sort and filter the data. You can also use parameters with data sources to specify criteria for which data should be retrieved, deleted, or updated. See the section "Updating data with the DataSet Designer," later in this chapter, for an introduction to working with `DataSet`s.

You can configure data sources by using the databound controls.

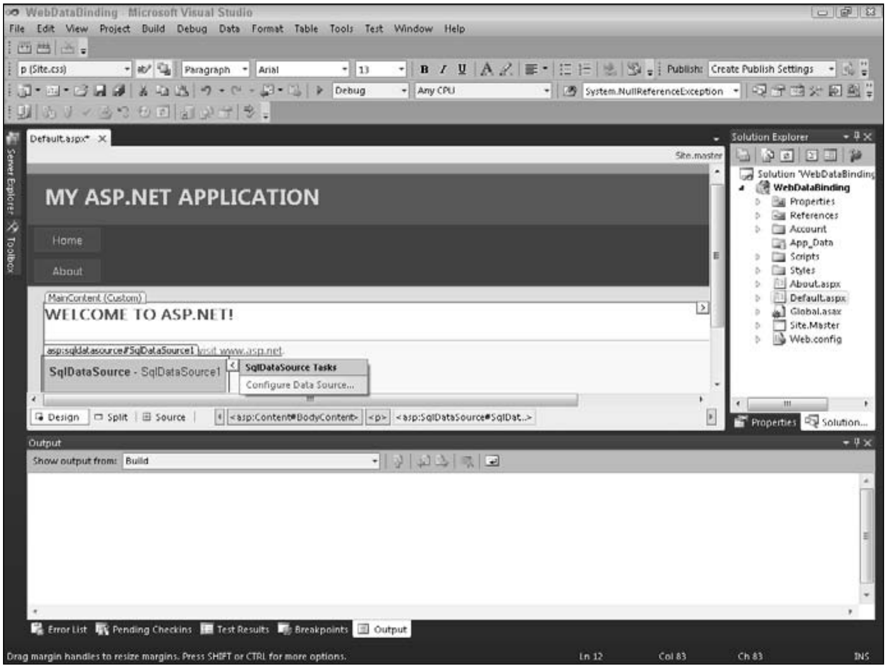


Figure 2-14: Click the Configure Data Source link to launch a configuration wizard.

Using databound Web server controls

The ASP.NET databound controls are bound to a data source control. Visual Studio provides built-in designer support for configuring data sources to work with databound controls.

The steps for adding databound controls are similar for all the controls:

1. **Drag and drop a databound control, such as `DataList`, onto the Web Form.**

   Databound controls are grouped on the Data tab of the toolbox.

2. **Right-click the arrow in the upper-right corner of the control to display the control's Tasks dialog box.**

3. **Select New Data Source from the Choose Data Source drop-down list, as shown in Figure 2-15.**

   The Data Source Configuration Wizard appears.

   Alternatively, you can select an existing data source control. See the earlier section "Going to the source" for more information on creating data source controls.

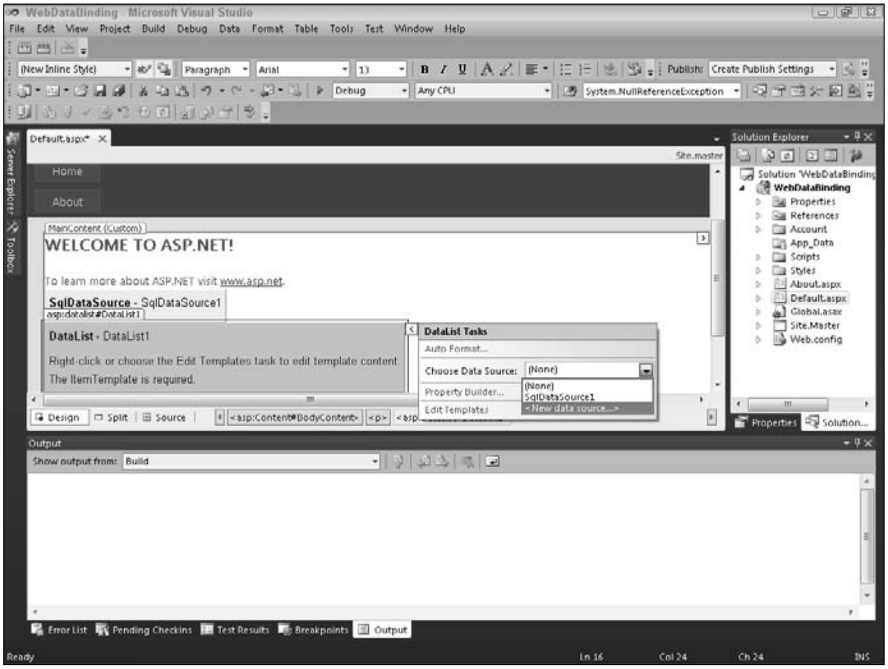


Figure 2-15: Select New Data Source to add a new data source.

You associate an existing data source control with a databound control using the databound control `DataSourceID` property.

4. **Click the data source icon on the Choose a Data Source Type page.**
5. **Click Next.**

   Depending on the data source you choose, the wizard displays a dialog box for configuring the data source.

   This step displays the same wizard you see after completing Step 3.

   After you configure the data source, Visual Studio adds a data source control to match the type of data source you select on the first page of the Data Source Configuration Wizard.
6. **Step through the wizard to configure the data source.**

Customizing layout with templates

All Web server controls are converted to HTML markup so they can be displayed in a Web browser. Some controls, such as the `GridView` control and `DetailsView` control, are configured by default to render as HTML tables. Other controls, such as `FormView` and `Repeater`, allow you to specify the HTML markup you wish to use to display the control's contents.

You specify the HTML markup that determines how a control is displayed by setting the control's template properties. (The content you define in a template property comprises a *template.*) You can define several template properties for a control, such as

- `HeaderTemplate`: Defines the HTML markup used to create the control's header.
- `ItemTemplate`: Defines the HTML markup used to display data items. Several variations of the `ItemTemplate` allow you to specify different HTML markup for alternating item rows or rows being edited. By specifying different markup, you provide visual cues to your user.
- `PagerTemplate`: Defines the HTML markup used for navigating between pages as data. For example, you might specify to use Next and Previous.
- `FooterTemplate`: Defines the HTML markup used to create the control's footer.

Data controls aren't the only controls that use template properties. Many Web server controls allow you to define their layouts by using templates. See the topic "ASP.NET Web Server Controls Templates" in the Visual Studio documentation for more information.

By using templates, you can explicitly define the HTML markup used to display your data in the Web browser. You can display data from a data source using controls, such as labels and text boxes, that are databound to the data source. The following example shows you how to create an `AlternatingItemTemplate` property for a `DataList` control:

1. **Drag and drop a data source control, such as a `SqlDataSource` control, on a Web Forms page**.
2. **Configure the data source control to access a data source, such as the Department table in the Adventure Works sample database.**

   See the earlier section "Going to the source" for more information about adding data sources to your Web pages.
3. **Drag and drop a `DataList` control on a Web Forms page.**
4. **Choose the data source you create in Steps 1 and 2 from the Choose Data Source drop-down list.**

   The DataList `ItemTemplate` property updates with the fields from the data source.
5. **Click the EditTemplates link in the Common DataList Tasks window.**

   The control appears in Template Editing Mode, as shown in Figure 2-16.

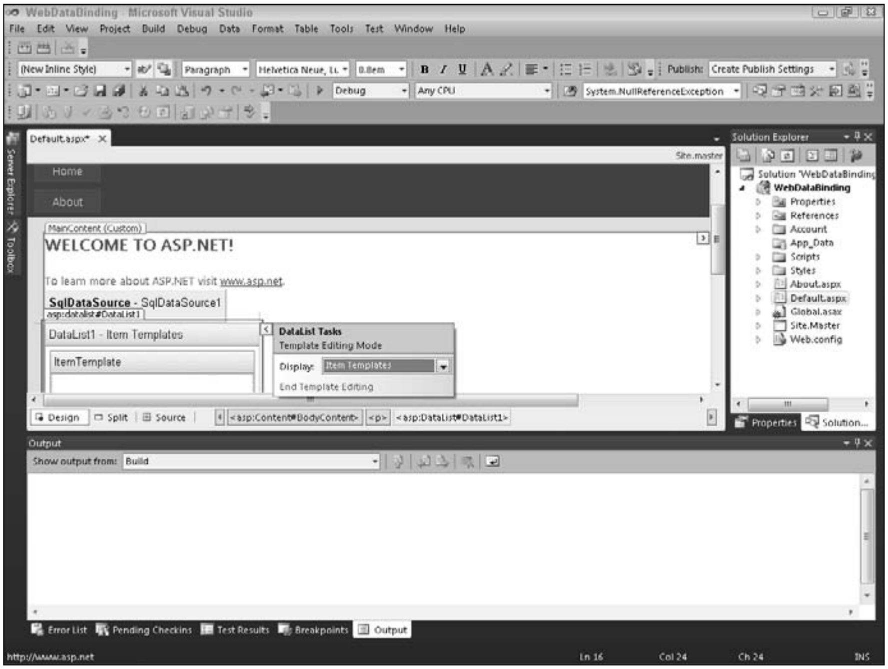


Figure 2-16: Customize the template using Template Editing Mode.

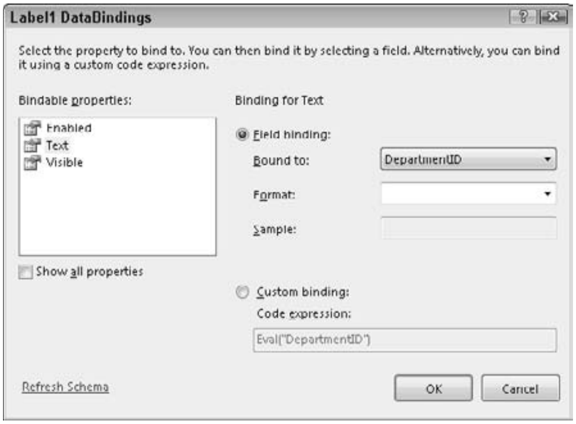
6. **Choose AlternatingItemTemplate from the Display drop-down list in the Common DataList Tasks window.**

   The `AlternatingItemTemplate` appears in the control.

7. **Drag and drop a `Label` control inside the `AlternatingItemTemplate` and then click the EditDataBindings link in the Label Tasks window.**

   The DataBindings dialog box appears.

8. **Select the Field Binding radio button and then choose a field from the Bound To drop-down list, as shown in Figure 2-17. Then click OK.**



Label1 DataBindings

Select the property to bind to. You can then bind it by selecting a field. Alternatively, you can bind it using a custom code expression.

Bindable properties: Enabled, Text, Visible

□ Show all properties

Binding for Text

● Field binding:

Bound to: DepartmentID

Format:

Sample:

○ Custom binding:

Code expression: Eval("DepartmentID")

Figure 2-17: Bind your data to the control.

   If the Field Binding option isn't available, click the Refresh Schema link (lower left). After the schema is refreshed, you should be able to select the option. If the option doesn't become available, you have a problem with your data source.

   The `Label` control `Text` property is bound to the field you select. You can choose to bind the data to a different property in the DataBindings dialog box. When you run the page, data from the field appears in the `Label`.

   An alternative approach to manually adding content to the `Alternative ItemTemplate` property is to copy and paste the content from the `ItemTemplate` property.

9. **Repeat Steps 7 and 8 to add all the data you wish to display from your data source.**

You now have a `DataList` control that has templates defined for the `ItemTemplate` and `AlternatingItemTemplate` properties. In order to provide a visual cue to the user for the alternating rows, you must define the style properties for the template. Each template has its own set of style properties. You use the style properties to create a custom look for displaying your data.

To set the style for a template in a Web server control:

1. **Click the control in the Web Forms Designer to select it.**

   For example, click the `DataList` control used in the preceding example.

2. **Press F4 to open the Properties window.**

3. **Scroll to the Styles properties.**

   If you have properties grouped using the Categorized button in the Properties window, all the style properties are grouped together in the Styles category. All the style properties end in the word *style,* such as `HeaderStyle` or `ItemStyle`. Each style corresponds to a template.

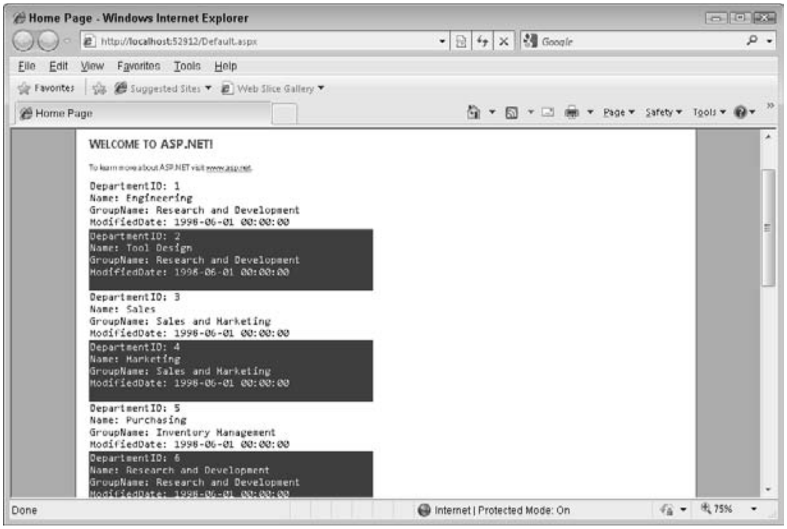
4. **Expand the style property you wish to set.**

   For example, expand the `AlternatingItemStyle` property.

5. **Set the individual property items or specify a `CssClass` you wish to use from a CSS.**

   For example, select a `BackColor` for the `AlternatingItemStyle` property to set a different background color for alternating items.

Figure 2-18 shows the `DataList` control with the `BackColor` property set for the `AlternatingItemStyle` property. The data displayed in the alternating rows uses the `AlternatingItemTemplate` property defined earlier in this section.


Home Page - Windows Internet Explorer
http://localhost:52912/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page Safety Tools
Done
Internet | Protected Mode: On
75%

WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

```
DepartmentID: 1
Name: Engineering
GroupName: Research and Development
ModifiedDate: 1998-06-01 00:00:00
DepartmentID: 2
Name: Tool Design
GroupName: Research and Development
ModifiedDate: 1998-06-01 00:00:00
DepartmentID: 3
Name: Sales
GroupName: Sales and Marketing
ModifiedDate: 1998-06-01 00:00:00
DepartmentID: 4
Name: Marketing
GroupName: Sales and Marketing
ModifiedDate: 1998-06-01 00:00:00
DepartmentID: 5
Name: Purchasing
GroupName: Inventory Management
ModifiedDate: 1998-06-01 00:00:00
DepartmentID: 6
Name: Research and Development
GroupName: Research and Development
ModifiedDate: 1998-06-01 00:00:00
```

Figure 2-18: Use styles to control the appearance of your data.

Working with tasks and properties

Databound Web server controls share many common tasks and properties. Table 2-1 lists some common tasks and properties. Tasks are accessed by using the control's Tasks dialog box, and properties are accessed by way of the Properties window.

Table 2-1 Common Data Web Server Control Tasks and Properties

Feature	*How to Access It*	*Controls*
Choose Data Source	Tasks	All data Web server controls
Auto Format	Tasks	`DataList`, `GridView`, `DetailsView`, `FormView`
Edit Templates	Tasks	`DataList`, `GridView`, `DetailsView`, `FormView`
Edit Columns, Edit Fields	Tasks	`GridView`, `DetailsView`
Gridlines	Property	`DataList`, `GridView`, `DetailsView`
ShowHeader, ShowFooter	Property	`DataList`, `GridView`
Paging	Property	`GridView`, `DetailsView`, `FormView`

Use tasks and properties to specify style qualities, such as formatting and pagination. You can also determine whether a control allows data manipulation if the data source supports it.

Staying on the same page

Paging is a common property that wires a data display so that users can page through it. To change the paging properties for a `FormView` control, follow these steps:

1. **Add a `FormView` control to a Web page and configure it to access a data source, such as the Department table in the Adventure Works database.**

   See Chapter 1 in this mini-book for information about installing the Adventure Works sample database from Microsoft.

2. **In the FormView Tasks dialog box, set the Enable Paging checkbox.**

3. **In the Properties window, click the plus sign (+) next to PagerSettings.**

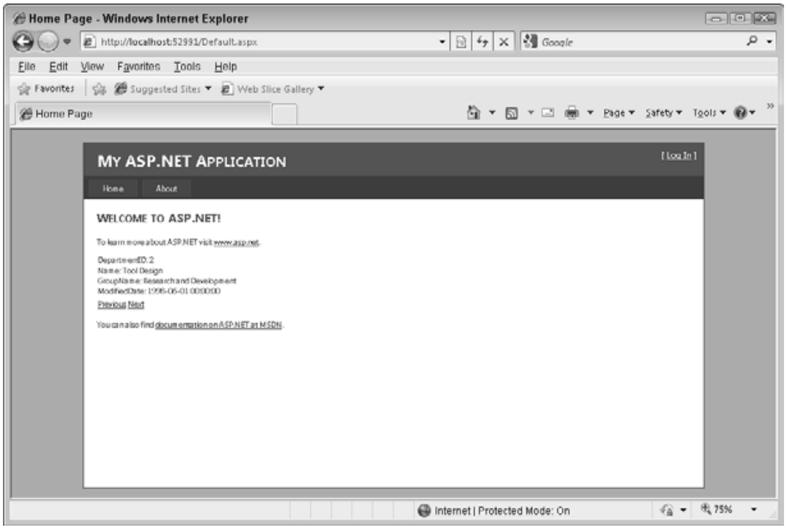
   The pager properties appear.

4. **Set these properties:**

   - `Mode`: Select NextPrevious.
   - `NextPageText`: Type **Next**.
   - `PreviousPageText`: Type **Previous**.

5. **Press Ctrl+F5 to run your site.**

   The pager displays the words `Previous` and `Next`, as shown in Figure 2-19.


Home Page - Windows Internet Explorer
http://localhost:52991/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page Safety Tools
My ASP.NET Application
[Log In]
Home
About
Internet | Protected Mode: On
75%

WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

DepartmentID: 2
Name: Tool Design
GroupName: Research and Development
ModifiedDate: 1998-06-01 00:00:00

Previous Next

You can also find documentation on ASP.NET at MSDN.

Figure 2-19: Use the `PagerSettings` property to set custom data paging settings.

AutoFormatting

All the databound Web controls can be customized. The AutoFormat feature is an easy way to apply a custom style to databound controls. To apply AutoFormatting to a control, follow these steps:

1. **In the FormView Tasks dialog box, select the Auto Format option.**

   The Auto Format dialog box appears.

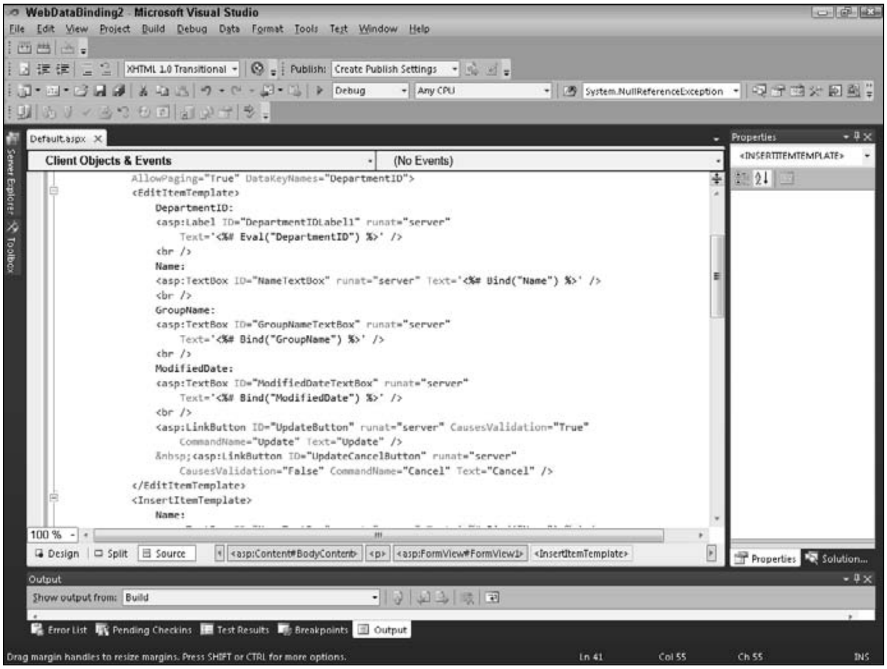
2. **Select a format from the list of schemes.**

   The selected format is previewed in the Preview pane.

3. **Click OK.**

   The format is applied to the control.

To customize the Auto Format style, edit the control's `Style` properties in the Properties window or in the page's Source view, as shown in Figure 2-20.



```
AllowPaging="True" DataKeyNames="DepartmentID">
<EditItemTemplate>
    DepartmentID:
    <asp:Label ID="DepartmentIDLabel1" runat="server"
        Text='<%# Eval("DepartmentID") %>' />
    <br />
    Name:
    <asp:TextBox ID="NameTextBox" runat="server" Text='<%# Bind("Name") %>' />
    <br />
    GroupName:
    <asp:TextBox ID="GroupNameTextBox" runat="server"
        Text='<%# Bind("GroupName") %>' />
    <br />
    ModifiedDate:
    <asp:TextBox ID="ModifiedDateTextBox" runat="server"
        Text='<%# Bind("ModifiedDate") %>' />
    <br />
    <asp:LinkButton ID="UpdateButton" runat="server" CausesValidation="True"
        CommandName="Update" Text="Update" />
    &nbsp;<asp:LinkButton ID="UpdateCancelButton" runat="server"
        CausesValidation="False" CommandName="Cancel" Text="Cancel" />
</EditItemTemplate>
<InsertItemTemplate>
    Name:
```

Figure 2-20: Customize the Auto Format style by using the page's Source view.

Updating data with the DataSet Designer

Visual Studio provides the DataSet Designer as a means to add data to your Web site quickly and easily. The DataSet Designer makes light work of these tasks:

- ✦ Creating data sources
- ✦ Retrieving data from data sources
- ✦ Creating commands to update data in data sources

The `DataSet` stores an offline copy of the data retrieved from a data source.

To create a new `DataSet`, follow these steps:

1. **Click the Web site's project folder in Solution Explorer and then choose Website➪Add New Item.**

   The Add New Item dialog box appears.

2. **Click the DataSet icon and enter a name, such as** AdventureWorks.xsd, **in the Name text box. Then click Add.**

   Visual Studio prompts you to add a new App_Code folder.

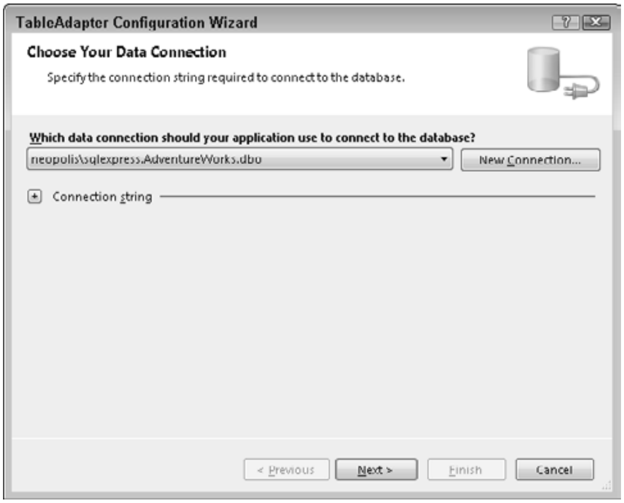
3. **Click Yes.**

4. **Drag a `TableAdapter` from the toolbox onto the AdventureWorks.xsd page.**

   The TableAdapter Configuration Wizard appears.

5. **Select an existing database connection or create a new connection on the Choose Your Data Connection page in the wizard.**

   If you already created a connection to the Adventure Works database by following earlier examples, you can reuse that connection by choosing it from the drop-down list of connections. Figure 2-21 shows the wizard reusing an existing connection.



TableAdapter Configuration Wizard

Choose Your Data Connection

Specify the connection string required to connect to the database.

Which data connection should your application use to connect to the database?

neopolis\sqlexpress.AdventureWorks.dbo

Figure 2-21: Select an existing database connection or create a new one.

   Click the New Connection button to create a new connection to the Adventure Works database. See the earlier section "Connecting to databases" to see how to configure a new database connection.

6. **Click Next.**

   The Choose a Command Type page appears.

7. **Accept the default command choice, Use SQL Statements. Then click Next.**

   You may use this page in the wizard to choose new or existing stored procedures instead of SQL statements.

8. **On the Enter an SQL Statement page, click the Query Builder button.**

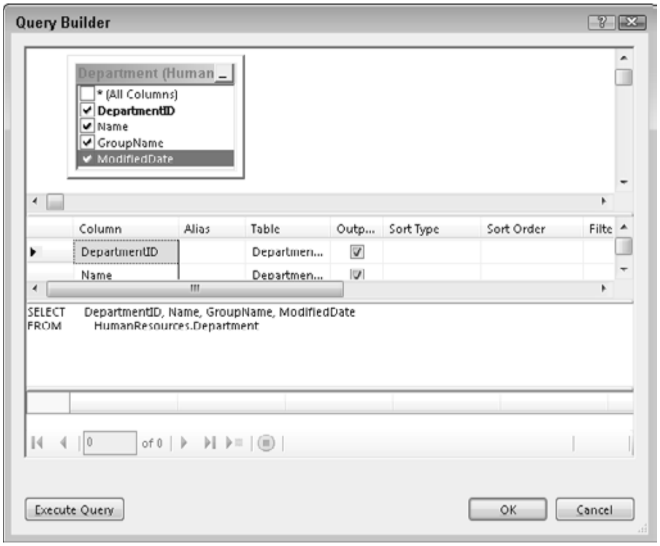
   The Query Builder appears with the Add Table dialog box open.

9. **Select a table, such as the Department table, and click the Add button to add the table to the Query Builder.**

10. **Click the Close button to close the Add Table window.**

11. **Select each column in the Department table to add the columns to the query's output, as shown in Figure 2-22.**

    You can select the asterisk-symbol (All Columns) check box to select all columns in the table, but selecting columns individually is usually considered best practice. If you use the All Columns option, you run the risk of displaying additional columns if a new column is added to the table in the database.



□ * (All Columns)
☑ DepartmentID
☑ Name
☑ GroupName
☑ ModifiedDate

	Column	Alias	Table	Outp...	Sort Type	Sort Order	Filte
▶	DepartmentID		Departmen...	☑			
	Name		Departmen...	☑			

```
SELECT    DepartmentID, Name, GroupName, ModifiedDate
FROM        HumanResources.Department
```

Figure 2-22: Select all the columns in the Department table for your query.

12. **Click the Execute Query button to test your query and then click OK to close the Query Builder.**

    Your SQL statement appears in the TableAdapter Configuration Wizard.

13. **Click Next.**

    The Choose Methods to Generate page appears.

14. **Accept the defaults and click Next.**

    The Wizard Results page appears.

15. **Click Finish.**

Visual Studio creates the DataSet and a TableAdapter for accessing and updating the Department table. See Chapter 3 in this mini-book for more information about DataSets and TableAdapters.

To use the DataSet as a data source for a data source Web control, follow these steps:

1. **Drag and drop an `ObjectDataSource` control on the Web Form.**
2. **Right-click the arrow in the upper-right corner to display the control's Tasks dialog box.**
3. **Click the Configure Data Source link.**

   The Configure Data Source Wizard appears.
4. **Select the `TableAdapter` from the Business Object drop-down list.**

   If you used the Department table from the Adventure Works database, the name of the TableAdapter is DepartmentTableAdapter.
5. **Click Next.**

   The Define Data Methods page appears. The methods created by the DataSet Designer appear on the SELECT, UPDATE, INSERT, and DELETE tabs. You may use the default methods or select different methods.

   To create new methods, right-click the TableAdapter in the DataSet Designer and then choose Configure. The TableAdapter Configuration Wizard starts, and you can specify methods to generate.
6. **Click Finish.**

   You have a DataSource control that you can use as the data source for a databound control.

DataSets are reusable. To add an existing DataSet from another project to your project, right-click the project's folder and choose Add Existing Item.

Getting the view right with the GridView control

By far, the easiest data Web server controls to use are GridView and DetailsView. They have built-in support for sorting, paging, selecting, editing, and deleting.

Sorting and paging

To enable sorting and paging on a GridView control, follow these steps:

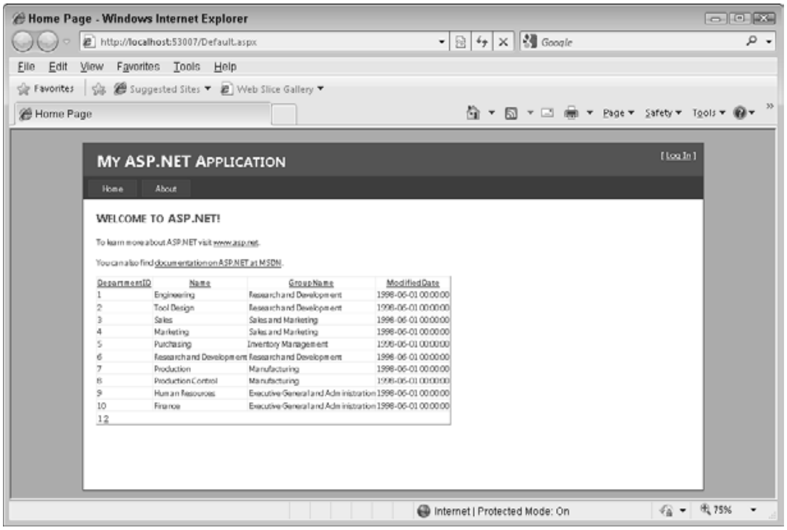
1. **Right-click the upper-right corner of the control to display the control's Tasks dialog box.**
2. **Select the Enable Paging and Enable Sorting options.**

3. **Press Ctrl+F5 to run your site.**

   The data appears in a table.

4. **To sort the table by the column — such as GroupName — click the column name hyperlink.**

5. **To page forward, click the numeral 2 in the table's footer (the little underlined 2 shown in Figure 2-23).**


Home Page - Windows Internet Explorer
http://localhost:53007/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page Safety Tools
MY ASP.NET APPLICATION
[Log In]
Home
About
Internet | Protected Mode: On
75%

WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

You can also find documentation on ASP.NET at MSDN.

DepartmentID	Name	GroupName	ModifiedDate
1	Engineering	Research and Development	1998-06-01 00:00:00
2	Tool Design	Research and Development	1998-06-01 00:00:00
3	Sales	Sales and Marketing	1998-06-01 00:00:00
4	Marketing	Sales and Marketing	1998-06-01 00:00:00
5	Purchasing	Inventory Management	1998-06-01 00:00:00
6	Research and Development	Research and Development	1998-06-01 00:00:00
7	Production	Manufacturing	1998-06-01 00:00:00
8	Production Control	Manufacturing	1998-06-01 00:00:00
9	Human Resources	Executive General and Administration	1998-06-01 00:00:00
10	Finance	Executive General and Administration	1998-06-01 00:00:00
1 2			

Figure 2-23: Click the hyperlink to the table's footer to advance to the table's next page.

Editing and deleting

When `GridView` uses an updatable data source, such as a `DataSet`, you can use the built-in functionality of `GridView` for editing and deleting records. To enable editing and updating in a `GridView` control, follow these steps:

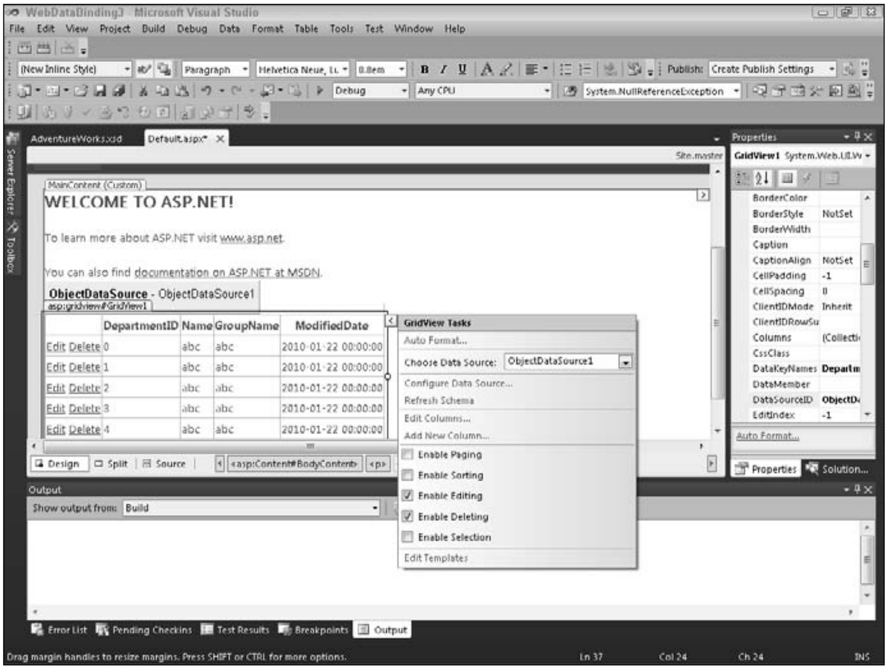
1. **Drag and drop a `GridView` control on the form.**

2. **Set the GridView data source to an updatable data source, such as the `ObjectDataSource` control you create in the section "Updating data with the DataSet Designer," earlier in this chapter.**

   The list of available tasks expands.

3. **Select the Enable Editing and the Enable Deleting options, as shown in Figure 2-24.**

   The Edit and Delete commands are added to the `GridView` control.



WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

You can also find documentation on ASP.NET at MSDN.

ObjectDataSource - ObjectDataSource1

asp:gridview#GridView1

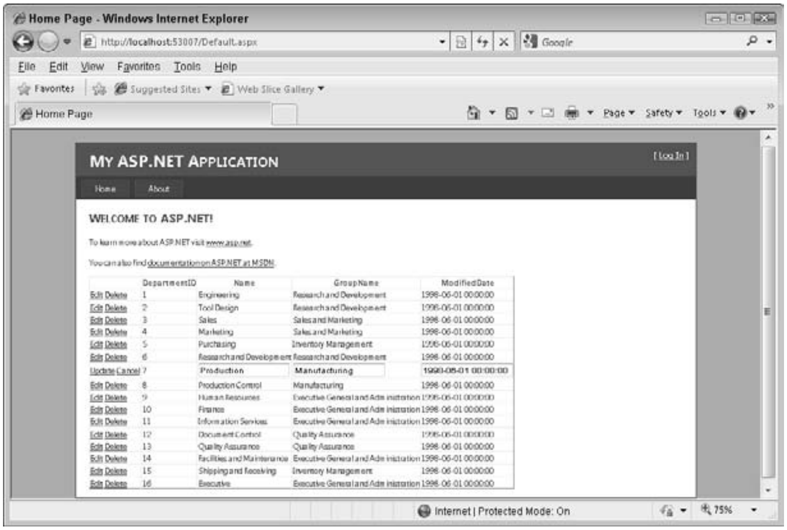
	DepartmentID	Name	GroupName	ModifiedDate
Edit Delete	0	abc	abc	2010-01-22 00:00:00
Edit Delete	1	abc	abc	2010-01-22 00:00:00
Edit Delete	2	abc	abc	2010-01-22 00:00:00
Edit Delete	3	abc	abc	2010-01-22 00:00:00
Edit Delete	4	abc	abc	2010-01-22 00:00:00

GridView Tasks
Auto Format...
Choose Data Source: ObjectDataSource1
Configure Data Source...
Refresh Schema
Edit Columns...
Add New Column...
□ Enable Paging
□ Enable Sorting
☑ Enable Editing
☑ Enable Deleting
□ Enable Selection
Edit Templates

Figure 2-24: Select Enable Editing and Enable Deleting.

4. **Press Ctrl+F5 to run the site.**
5. **Click the Edit button on one of the rows.**

   The row is placed in Edit mode, as shown in Figure 2-25.


Home Page - Windows Internet Explorer
http://localhost:53007/Default.aspx
Google
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Web Slice Gallery
Home Page
Page
Safety
Tools
MY ASP.NET APPLICATION
[Log In]
Home
About
Internet | Protected Mode: On
75%

WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

You can also find documentation on ASP.NET at MSDN.

	DepartmentID	Name	GroupName	ModifiedDate
Edit Delete	1	Engineering	Research and Development	1998-06-01 00:00:00
Edit Delete	2	Tool Design	Research and Development	1998-06-01 00:00:00
Edit Delete	3	Sales	Sales and Marketing	1998-06-01 00:00:00
Edit Delete	4	Marketing	Sales and Marketing	1998-06-01 00:00:00
Edit Delete	5	Purchasing	Inventory Management	1998-06-01 00:00:00
Edit Delete	6	Research and Development	Research and Development	1998-06-01 00:00:00
Update Cancel	7	Production	Manufacturing	1998-06-01 00:00:00
Edit Delete	8	Production Control	Manufacturing	1998-06-01 00:00:00
Edit Delete	9	Human Resources	Executive General and Administration	1998-06-01 00:00:00
Edit Delete	10	Finance	Executive General and Administration	1998-06-01 00:00:00
Edit Delete	11	Information Services	Executive General and Administration	1998-06-01 00:00:00
Edit Delete	12	Document Control	Quality Assurance	1998-06-01 00:00:00
Edit Delete	13	Quality Assurance	Quality Assurance	1998-06-01 00:00:00
Edit Delete	14	Facilities and Maintenance	Executive General and Administration	1998-06-01 00:00:00
Edit Delete	15	Shipping and Receiving	Inventory Management	1998-06-01 00:00:00
Edit Delete	16	Executive	Executive General and Administration	1998-06-01 00:00:00

Figure 2-25: Click the Edit button to place the row in Edit mode.

6. **Modify the record and then click Update (on the far left of the row).**

   The record is updated.

7. **Click the Delete button.**

   You see an error message. The Delete button tries to delete the record, but the database doesn't let it. This example shows why you have to make sure that the data source can execute the commands that Visual Studio builds.

Generally speaking, *delete* is a bad word when you're working with data. You should mark records as inactive rather than delete them. This strategy preserves the integrity of the data.

Binding to expressions

Visual Studio allows you to bind a control's property to an expression. You can bind a property to these kinds of expressions:

- `AppSettings`
- `ConnectionStrings`
- `Resources`

`AppSettings` and `ConnectionStrings` expressions are stored in the `web.config` file. *Resources* are resource files that store user interface information. To bind a control's property to an expression, you must first create something to which to bind. Book III, Chapter 6 walks you through how to add a name-value pair to the `AppSettings` section of the `web.config` file.

To bind to an `AppSetting` key named `AppName`, follow these steps:

1. **Select a data control and view the control's properties by right-clicking and choosing Properties from the contextual menu.**

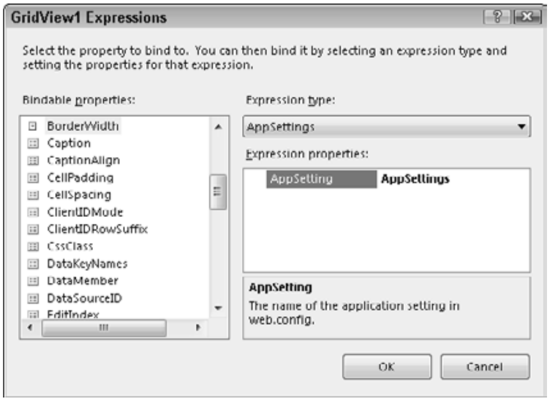
2. **In the Properties window, click the ellipsis button for the `Expressions` property.**

   The Expressions dialog box appears.

3. **Select a property from the list of bindable properties.**

4. **Select AppSettings from the Expressions Type drop-down list.**

5. **Type** AppSettings **in the Expressions Properties box, as shown in Figure 2-26.**



GridView1 Expressions

Select the property to bind to. You can then bind it by selecting an expression type and setting the properties for that expression.

Bindable properties:

Expression type: AppSettings

Expression properties:

AppSetting AppSettings

AppSetting
The name of the application setting in web.config.

Figure 2-26: Type an `App-Settings` key in the properties box.

6. **Click OK.**

When you run the site, the property's value is replaced with the value that the `AppSettings` expression evaluates to.

Chapter 3: Working with Strongly Typed DataSets

In This Chapter

- **Creating DataSets in a class library**
- **Using the DataSet Designer**
- **Accessing data with TableAdapters**
- **Digging into the code of a typed DataSet**
- **Consuming typed DataSets**

Programmers have always needed a way to work with data in their applications. Accessing data from files or databases every time you need data is impractical. To accommodate the need to have programs access data quickly, programmers store data in memory, using everything from arrays to integers to custom data structures.

Instead of making you spend lots of time thinking about how to store data in memory, Microsoft created the DataSet to provide a standardized means of storing data in memory for quick access.

A DataSet works like an in-memory database. The DataSet has tables with columns and rows, and you create relations between tables. You can load data into a DataSet from a database, or you can enter new data into the DataSet. You can insert, delete, and update records. After you make all the changes to records in your DataSet, you commit those changes to the actual database.

Visual Studio provides the DataSet Designer for creating and manipulating DataSets. This chapter explains what a DataSet is and shows you how to use the DataSet Designer to create a special kind of DataSet, the strongly typed DataSet.

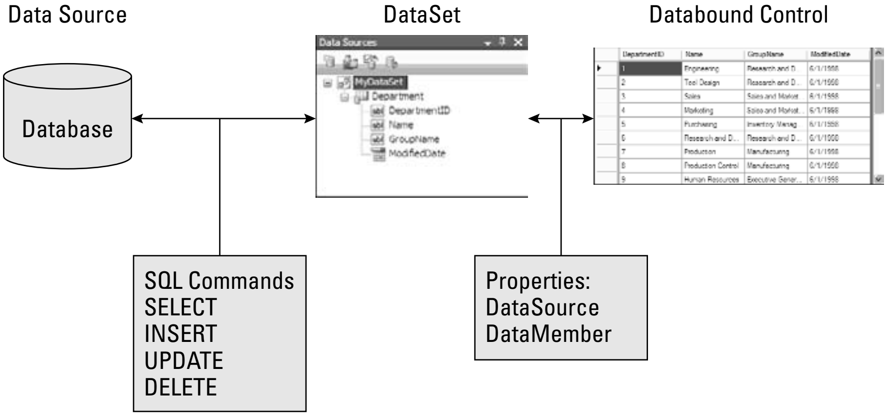
Understanding the DataSet

A DataSet is a resource provided by the Microsoft .NET Framework for storing data in memory. DataSets store data in tables, and you can create relations between tables, similar to how you create them in a database.

You use a DataSet to

- **Store an in-memory copy of data from a database or other data source.** Data stored in a DataSet is disconnected from its original data source. The DataSet stores data in the computer's memory, where your program can quickly access it.
- **Store data modified by your program.** As data stored in the DataSet is modified by your program or people using your program, the DataSet maintains copies of the original and modified records. You can specify constraints that prevent records from being modified or deleted in the DataSet. Because DataSets don't remain connected to the data source, you can provide feedback to your users without modifying the original data source.
- **Execute commands that retrieve data from the data source and send modified data back to the data source.** DataSets accept a set of SQL statements specified by the programmer to select, insert, update, and delete records in the data source.
- **Act as a data source for databound controls, such as GridView.** For more about this topic, you can check out Chapter 2 of Book V. DataSets are a natural data source for databound controls. You assign your DataSet and a table in the DataSet to the control's DataSource and DataMember properties, and the control displays the data from the DataSet.

The .NET DataSet acts as an intermediary between your data source and a databound control, as shown in Figure 3-1.



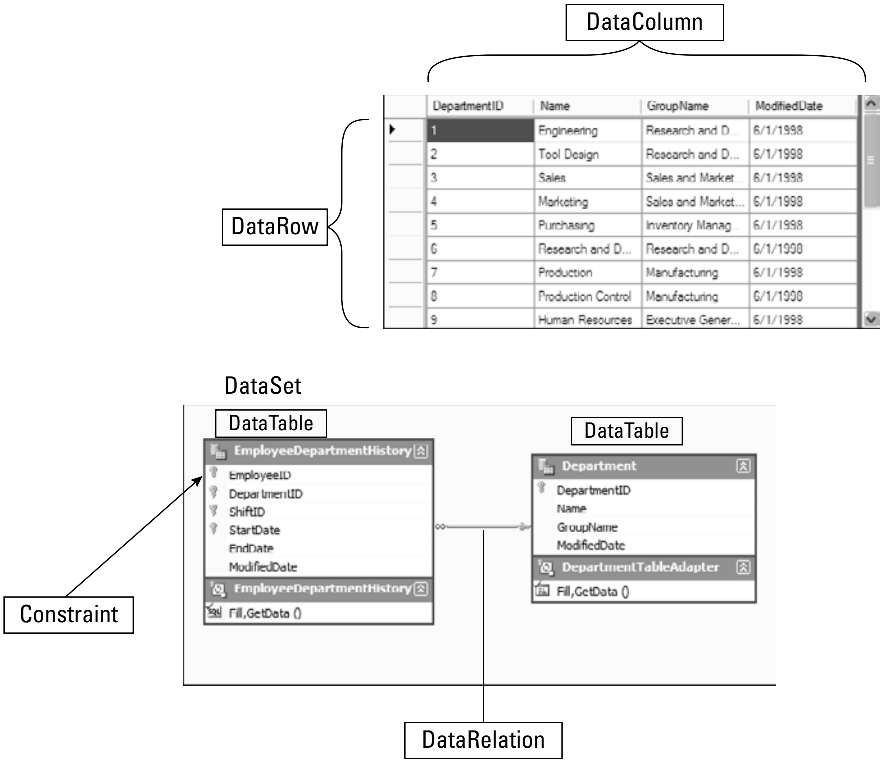
DepartmentID	Name	GroupName	ModifiedDate
1	Engineering	Research and D...	6/1/1998
2	Tool Design	Research and D...	6/1/1998
3	Sales	Sales and Market...	6/1/1998
4	Marketing	Sales and Market...	6/1/1998
5	Purchasing	Inventory Manag...	6/1/1998
6	Research and D...	Research and D...	6/1/1998
7	Production	Manufacturing	6/1/1998
8	Production Control	Manufacturing	6/1/1998
9	Human Resources	Executive Gener...	6/1/1998

Figure 3-1: A DataSet sits between a database and one or more databound controls.

The .NET DataSet, which provides a basic structure for storing data, is composed of a set of these objects:

- DataTable: Contains a set of DataColumn and DataRow objects that represent a tabular set of data
- DataColumn: Represents a column in a DataTable object
- Constraint: Identifies DataColumn objects as unique or foreign keys
- DataRelation: Creates a parent-child relationship between two DataTable objects
- DataRow: Represents a row in a DataTable object

A DataSet can contain one or more DataTables. A DataTable contains one or more DataColumn and DataRow objects. Figure 3-2 illustrates the relationship between the objects in a DataSet.



DepartmentID	Name	GroupName	ModifiedDate
1	Engineering	Research and D...	6/1/1998
2	Tool Design	Research and D...	6/1/1998
3	Sales	Sales and Market...	6/1/1998
4	Marketing	Sales and Market...	6/1/1998
5	Purchasing	Inventory Manag...	6/1/1998
6	Research and D...	Research and D...	6/1/1998
7	Production	Manufacturing	6/1/1998
8	Production Control	Manufacturing	6/1/1998
9	Human Resources	Executive Gener...	6/1/1998

Figure 3-2: A DataSet contains a set of DataTables.

Discerning typed versus untyped DataSets

You can create two kinds of `DataSets` in .NET:

- **Typed:** This kind uses a schema from an existing data source to define the tables, columns, and constraints in the `DataSet`.
- **Untyped:** No schema is defined when the `DataSet` is created.

Because no schema is defined for untyped `DataSet`s, you must create each `DataTable` you need, along with the `DataColumn` objects. A typed `DataSet` is an intelligent `DataSet` because it's "aware" of an existing schema. Visual Studio provides the DataSet Designer visual tool for creating typed `DataSet`s.

Typed `DataSet`s force you to use the types defined in the `DataSet` schema. For example, a typed `DataSet` that uses a schema to store customer data can't be used to store order data. You must modify the `DataSet` schema so that the `DataSet` knows how to store order data. An untyped `DataSet`, however, has no such restriction. You can store customer and order data in an untyped `DataSet` without explicitly defining how the data is stored.

Although it may be tempting to think that using a `DataSet` is easier if you didn't have to explicitly define a schema, using untyped `DataSet`s allows you to get sloppy and can make your code difficult to read and follow. Using typed `DataSet`s eliminates the ambiguity of not knowing for sure what data is stored in the `DataSet`. By referring to the `DataSet` schema, you have no doubts about the typed `DataSet` purpose in your code. The DataSet Designer makes creating typed `DataSet`s very simple.

Typed `DataSet`s are often called *strongly typed,* and untyped `DataSet`s are usually just referred to as `DataSet`s.

What's your type?

Typed and untyped `DataSet`s are nothing new to .NET and Visual Studio. In early versions of .NET, hardly anyone used typed `DataSet`s. The code generated for typed `DataSet`s by previous versions of Visual Studio had some quirks. People who saw how easy it was to use typed `DataSet`s figured out ways to work around the flaws. Microsoft has worked out the bugs in the last release of Visual Studio and .NET. Microsoft is again singing the praises of typed `DataSet`s, although most articles you'll see about typed `DataSet`s refer to them as *strongly typed `DataSets`.* Whether you choose to call them *typed* or *strongly typed,* using them makes your coding go a lot faster.

Working with untyped DataSets

You create untyped DataSets by using the DataSet control on the Data tab of the control toolbox.

To create an untyped DataSet, follow these steps:

1. **Create a new Windows application.**
2. **Drag and drop a DataSet control from the Data tab of the toolbox onto the Windows Forms Designer.**

   The Add DataSet window appears.
3. **Select the Untyped Dataset option.**
4. **Click OK.**

   Visual Studio adds the DataSet to your form.

You can create a DataSet in code. (See Chapter 5 of Book V.)

An untyped DataSet has no schema. A schema defines the structure of the DataSet. You must add the tables and columns to define the schema. To add tables and columns to your untyped DataSet, follow these steps:

1. **Click the DataSet you added to your form in the preceding set of steps.**
2. **Press F4 to display the Properties window.**
3. **Click the ellipsis button for the Tables property.**

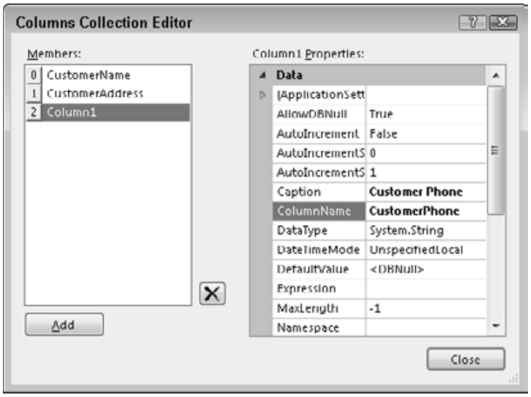
   The Tables Collection Editor appears.
4. **Click Add to add a new table.**

   The table's properties appear on the right side of the editor.
5. **Type a name for your table in the TableName property, such as Customer.**

   The table's default name is Table1.
6. **Click the ellipsis button for the Columns property.**

   The Columns Collection Editor appears.
7. **Click Add to add a new column.**
8. **Type a name for the column in the ColumnName property, such as CustomerName.**

   You can set the column's properties, such as its default value and maximum length, in the Columns Collection Editor (see Figure 3-3).
9. **Repeat Steps 3 through 8 to add more tables and columns.**


Columns Collection Editor

Members:

0 CustomerName
1 CustomerAddress
2 Column1

Column1 Properties:

Data	
(ApplicationSett	
AllowDBNull	True
AutoIncrement	False
AutoIncrementS	0
AutoIncrementS	1
Caption	**Customer Phone**
ColumnName	**CustomerPhone**
DataType	System.String
DateTimeMode	UnspecifiedLocal
DefaultValue	<DBNull>
Expression	
MaxLength	-1
Namespace	

Figure 3-3: Use the Columns Collection Editor to access the DataSet properties and collections.

You can use the DataSet Properties window to access collection editors for relations between tables and constraints within a table.

Using the Properties window to add tables and columns doesn't change your DataSet from untyped to typed. In order to be a typed DataSet, the DataSet must have a schema file.

Flexing your strongly typed DataSets

Data types are used in programming languages to set the rules for how data is expected to behave. Examples of data types are integers and characters. When you declare a new variable, you specify the data type of the variable so that your program knows what to expect. A strongly typed DataSet creates a new data type for your application to use.

Strongly typed programming languages require you to specify a data type at the time you declare a variable. *Weakly typed* languages also use data types, but they aren't as strict about their usage. For example, a weakly typed language allows you to treat a string as an integer. Strongly typed languages are stricter about how type rules are enforced. Because strongly typed languages allow you to bend fewer rules, the language's compiler has more opportunities to prevent you from using types incorrectly. Many scripting languages, such as JavaScript and Perl, are weakly typed. Visual Basic and C# are strongly typed languages. The computer doesn't actually care about data typing: All it sees is a series of bits. The use of data typing helps the developer work with the data correctly.

For example, if you create the typed DataSet CustomerDataSet to store customer data, you create a new data type. Your new customer DataSet might include tables to store your customer's identity and contact data. You can declare a new MyCustomer variable that uses your customer DataSet type, like this:

VB:

```
Dim MyCustomer As New CustomerDataSet()
```

C#:

```
CustomerDataSet MyCustomer = new CustomerDataSet();
```

When you access your `MyCustomer` variable, it already knows how you want to define your customer's identity and contact data because it's of the type `CustomerDataSet`. For example, to add a new customer and set the caption for the account number column, you type

VB:

```
MyCustomer.Customer.AddCustomerRow("123456", "John", "Smith")
MyCustomer.Customer.AccountNumberColumn.Caption = "Account Number"
```

C#:

```
MyCustomer.Customer.AddCustomerRow("123456", "John", "Smith");
MyCustomer.Customer.AddCustomerRow("123456", "John", "Smith");
```

Using typed `DataSet`s saves you from doing a lot of coding because Visual Studio generates the code for you. All you have to do is drag and drop tables from your database onto the DataSet Designer. To read more about the DataSet Designer, see the upcoming section "Creating Typed DataSets."

Take another look at `DataSet`s if you wrote them off in the past. In previous versions of Visual Studio, the code that was generated for typed `DataSet`s was buggy and difficult to extend.

DataSets and XML, together at last

`DataSet`s can use XML as their data source *and* be converted to XML. The XML language is used to create self-describing data files. For example, you can read an XML file into a `DataSet`, modify the data in the `DataSet`, and then produce the modified data as XML output. Or, you can open data from a database in a `DataSet` and produce the data as XML output.

You have to be able to write code to read and write XML to `DataSet`s. No wizards are available to do it for you. See Chapter 5 in Book V to read more.

Typed `DataSet`s use an XML schema to define their tables, columns, relations, and constraints. See the section "Viewing the source," later in this chapter, for more about viewing a typed `DataSet` XML schema.

Creating Typed DataSets

Visual Studio provides the DataSet Designer for creating and working with strongly typed `DataSet`s. A `DataSet` is *strongly typed* if it's generated from an existing data source schema that outlines the `DataSet` structure. You don't have to create this schema yourself — you can drag and drop items from a database onto the DataSet Designer. The DataSet Designer builds the schema for you.

REMEMBER

Whether you're using the DataSet Designer in a Windows application, Web site project, or smart-device project, the DataSet Designer looks and acts the same.

Creating a DataSet in a class library

When you work with data, use the best practice of isolating your data code into separate projects so that you can reuse your data with other presentation layers. For example, if you create a data project that a Windows project and a Web site project can access, isolate your data in its own project so that you don't have to define it twice.

To use an existing `DataSet` in a Windows or Web project, see the upcoming section "Using a Typed DataSet."

You create a data project by using a class library. To create a new class library, follow these steps:

1. **Choose File➪New.**
2. **Choose Project from the submenu.**

   The New Project dialog box appears.
3. **Expand the C# project type.**

   You can create class libraries in other project types, such as Visual Basic, F#, and C++.
4. **Click the Class Library icon.**
5. **Type a name, such as DataLibrary, in the Name text box.**
6. **Click OK.**

   Visual Studio creates a new class library project.

The class library project template creates an empty class file named `class1.vb`. You can safely delete this file for this example. To delete the file, right-click the file in Solution Explorer and then choose Delete from the contextual menu.

Visual Studio includes a project item named `DataSet` that you use to add a typed `DataSet` to your project.

To add a typed `DataSet` to your class library, follow these steps:

1. **Click the class library's project folder in Solution Explorer.**

   The project folder is under the solution.

2. **Choose Project➪Add New Item.**

   The Add New Item dialog box appears.

3. **Click the DataSet icon.**

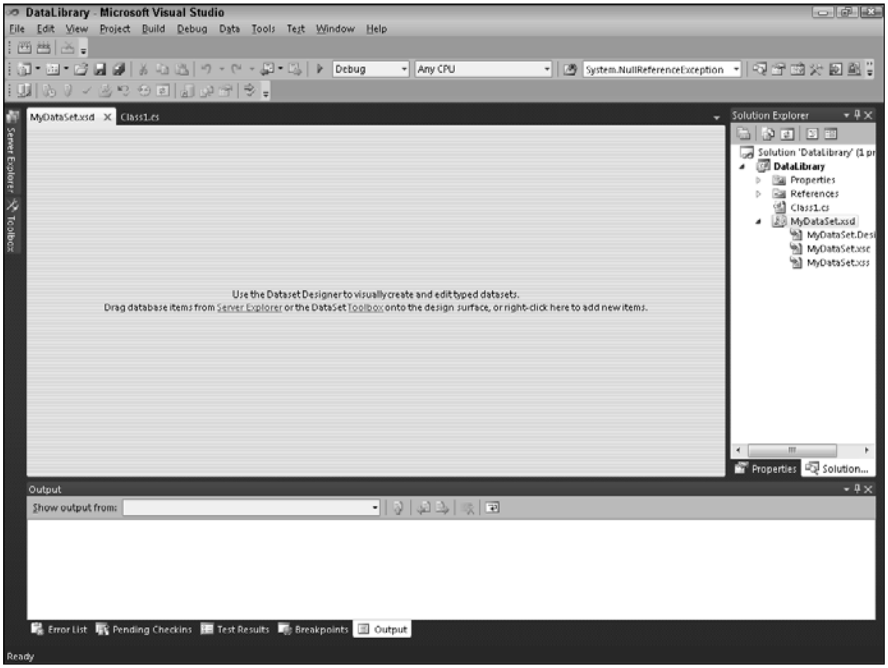
4. **Type a name, such as MyDataSet.**

   TIP

   Typed `DataSets` end with the file extension `.xsd`, which is the extension for XML schemas. To read more about XML schemas, see Chapter 4 in Book V.

5. **Click Add.**

   Visual Studio opens the typed `DataSet` in the DataSet Designer, as shown in Figure 3-4.



Use the Dataset Designer to visually create and edit typed datasets.
Drag database items from Server Explorer or the DataSet Toolbox onto the design surface, or right-click here to add new items.

Figure 3-4: Use the DataSet Designer to create typed `DataSets`.

Adding typed DataSets to existing projects

To access the DataSet Designer, you must create a new `DataSet` or open an existing one. The steps for these two tasks differ slightly, depending on whether you're opening a `DataSet` in a Web site project or a Windows project.

Follow these steps to add a strongly typed `DataSet` to a Web project:

1. **Right-click the Web site's project folder in Solution Explorer.**
2. **Choose Add New Item from the contextual menu.**

   The Add New Item dialog box appears.
3. **Click the DataSet icon.**
4. **Type a name for the `DataSet` in the Name text box.**
5. **Click Add.**

   Visual Studio prompts you to add the `DataSet` to the App_Code folder.

   The App_Code folder is part of the new Web site folder structure in ASP.NET 4. You can read more about the App_Code folder in Book IV, Chapter 4.
6. **Click Yes to add the `DataSet` to the Web site's App_Code folder.**

   The `DataSet` appears in the DataSet Designer.

Visual Studio launches the TableAdapter Configuration Wizard. The wizard allows you to create a connection to a database and configure SQL statements for selecting and updating data. To see the TableAdapter Configuration Wizard in action, see the section in Chapter 2 of Book V about updating data with the DataSet Designer.

To add a `DataSet` to a Windows project, follow these steps:

1. **Choose Project⇨Add New Item.**

   The Add New Item dialog box appears.
2. **Click the DataSet icon.**
3. **Type a name for the `DataSet` and then click Add.**

   The `DataSet` appears in the DataSet Designer.

To read more about the different ways to add `DataSets` to Windows applications, see Chapter 2 in Book V.

Exploring the DataSet Designer

Anytime you add a typed DataSet to your application, you use the DataSet Designer to create and modify its schema. A typed DataSet schema is stored in an XML Schema file with the file extension .xsd. The DataSet Designer is essentially a visual schema designer where you define the tables and relations in the typed DataSet. Visual Studio uses the schema to generate the code to create your typed DataSet.

The DataSet Designer provides two features for creating typed DataSets:

- **A toolbox:** Use the toolbox to build tables and relations.
- **A design surface:** Drag and drop items from the toolbox or an existing database connection to the design surface.

Meet the players

The DataSet Designer uses a distinctive set of toolbox items for building typed DataSets. When you drag and drop database objects (such as tables) from an existing database connection, Visual Studio populates items from the DataSet Designer toolbox by using the schema information from your database connection.

This list describes the toolbox items used by DataSet Designer:

- **TableAdapter:** Retrieves data from the database and sends updates back for an associated DataTable
- **Query:** Retrieves data from the database, not associated with a specific DataTable
- **DataTable:** Provides a representation of a database table
- **Relation:** Creates links and constraints between DataTables

The DataSet Designer toolbox items are visual representations of technologies used in *ADO.NET,* which is the data access technology in the Microsoft .NET Framework. To read more about ADO.NET, see Chapter 5 in Book V.

You use DataSet Designer to work with representations of your data visually. Behind the scenes, DataSet Designer

- Creates an XML schema that defines the tables and columns in your DataSet. The DataSet file itself is an XML schema file. DataSet Designer adds annotations to the XML schema files to extend the XML schema syntax.
- Generates the ADO.NET code that allows your program to access the data source.

See the section "Looking under the Hood of a Strongly Typed DataSet," later in this chapter, to read more about viewing the code and XML markup generated by DataSet Designer.

Building a DataSet of your own

You can use the tools in the DataSet Designer toolbox to manually create a typed `DataSet`.

Here are the steps you follow to add a new table to a typed `DataSet`:

1. **Create a new typed `DataSet` by following the steps in the earlier section "Creating Typed DataSets."**
2. **Drag and drop a `DataTable` from the toolbox onto DataSet Designer.**

   A `DataTable` with the name `DataTable1` appears in the DataSet Designer.

   Press Ctrl+Alt+X to open the toolbox if it's closed.
3. **Click the `DataTable` name and type a name, such as Customer.**

The `DataTable` needs columns. Add them by following these steps:

1. **Right-click the Customer `DataTable`.**
2. **From the contextual menu that appears, choose Add➪Column.**

   A new column appears in the `DataTable`.
3. **Type a name for the column, such as CustID.**
4. **Repeat Steps 1 through 3 to add FirstName and LastName columns to the `DataTable`.**

TIP

You can set properties for the `DataTable` columns by using the Properties window. Examples of properties you can set are the column's `DataType` and `DefaultValue` properties. See the section "Going beyond database tables," later in this chapter, for an example of setting a column's property.

You add constraints to your `Customer DataTable` to specify whether a column is unique. To set the CustID column as the table's primary key, follow these steps:

1. **Right-click the Customer `DataTable`.**
2. **From the contextual menu that appears, choose Add➪Key.**

   The Unique Constraint dialog box appears.
3. **Select the CustID column by placing a check mark next to it.**
4. **Select the primary key option.**

5. **Click OK.**

   A little key icon appears next to the CustID column name, as shown in Figure 3-5.

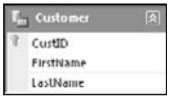


Figure 3-5: A key appears next to the CustID column name.

You use the `Relation` item from the DataSet Designer toolbox to create a parent-child relationship between two tables. The `Relation` item creates a relation that represents the relationship.

You can create a relationship between a table and itself. When a table is related to itself, the relationship is *reflexive.* See the next section for an example of a reflexive relationship.

You can create three kinds of relations by using the `Relation` item:

- **Both Relation and Foreign Key Constraint:** Creates a parent-child relationship between two tables by using the columns selected in the Columns grid. A foreign key constraint is created that allows you to set rules for what happens when records are updated or deleted.
- **Foreign Key Constraint Only:** Identifies columns in the child table as foreign keys and allows you to set constraints for what happens when records are updated or deleted. You can't access child records via their parent records because no parent-child relationship is established.
- **Relation Only:** Creates a parent-child relationship between the tables but doesn't implement rules for what happens when records are updated or deleted.

Select the Relation Only option whenever you want to access a child table's records via its parent's records. Create a foreign key constraint whenever you want to specify rules for when parent or child records are updated or deleted. Generally, set up both relation and foreign key constraints because this affords the highest amount of data integrity.

Follow these steps to create a relation between two tables:

1. **Follow the first two step lists in this section to create a `DataTable` and columns.**
2. **Create a `DataTable` named CustomerAddress and add these columns:**
   - CustID
   - Address1
   - Address2
   - City
   - State
   - Zip
3. **Drag and drop a `Relation` item from the toolbox onto the designer.**

   The Relation dialog box appears, which you use to specify a parent and child table, and select the columns in each table that link the tables.
4. **Type a name for the Relation in the Name field.**

   Relation names are usually some combination of the two tables' names, such as Customer_CustomerAddress.
5. **Select Customer as the parent table and CustomerAddress as the child table.**

   When your `DataSet` has only two tables, the DataSet Designer selects the parent-child tables for you.
6. **In the Columns grid, select the CustID column from each table.**

   The CustID column is the column that both columns share.

   Be sure to create columns before adding the relation. You can't add new columns by using the Columns grid.
7. **Select the Both Relation and Foreign Key Constraint options.**

   Choosing the Both Relation and Foreign Key Constraint option or the Foreign Key Constraint Only option enables the constraint rule selectors, as shown in Figure 3-6.
8. **Use these constraint rule selectors to set the foreign key constraints:**
   - *Update Rule*: Cascade
   - *Delete Rule*: Cascade
   - *Accept/Reject Rule*: Cascade

   Constraint rules specify the action taken on a child record by the database when its parent record is updated or deleted. Your choices for Update Rule and Delete Rule are the following:

- *None*: Nothing happens to the child rows when a parent is updated or deleted.
- *Cascade*: The child rows are deleted or updated.
- *SetNull*: The values in the child rows are set to null.
- *SetDefault*: The values in the child row are set to the default values.

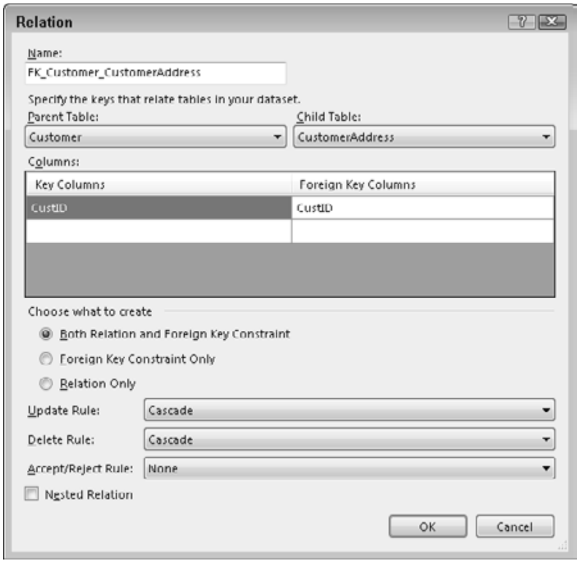
The *Accept/Reject Rule option* specifies what happens to the child rows when a parent row either accepts or rejects changes. Your choices for Accept/Reject Rule are

- *None*
- *Cascade*

9. **Leave the Nested Relation check box cleared.**

   Nesting the relation causes the child table to be nested inside the parent table.

10. **Click OK.**



Relation

Name: FK_Customer_CustomerAddress

Specify the keys that relate tables in your dataset.

Parent Table: Customer

Child Table: CustomerAddress

Columns:

Key Columns	Foreign Key Columns
CustID	CustID

Choose what to create

● Both Relation and Foreign Key Constraint

○ Foreign Key Constraint Only

○ Relation Only

Update Rule: Cascade

Delete Rule: Cascade

Accept/Reject Rule: None

□ Nested Relation

Figure 3-6: Choosing to create foreign key constraints enables the constraint rule selectors.

You use the `TableAdapter` and `Query` items from the DataSet Designer toolbox to fill your `DataSet` with data. Using the `TableAdapter` item opens the TableAdapter Configuration Wizard, and the `Query` item opens the TableAdapter Query Configuration Wizard. To see both wizards in action, see the section "Shepherding Data," later in this chapter.

Adding database objects

The easiest way to build a typed `DataSet` is to build it from an existing database. You use Server Explorer to drag and drop database objects onto the DataSet Designer surface. The designer presents a visual representation of the `DataSet` and creates a basic set of queries to select and update data. You then use a wizard to modify those queries or create new ones.

You use the set of Visual Database Tools, including Server Explorer, for manipulating database objects. See Chapter 6 in Book V for more information about the Visual Database Tools.

To connect to a database by using Server Explorer, follow these steps:

1. **Open Server Explorer by choosing View➪Server Explorer.**

   You can open Server Explorer by clicking the Server Explorer link from the DataSet Designer.

2. **Click the Connect to Database button on the Server Explorer toolbar.**

   The Add Connection dialog box appears.

   Alternatively, you can right-click the Data Connection icon and choose Add Connection from the contextual menu.

3. **Create a new connection to the Adventure Works database.**

   The connection appears in Server Explorer. (Chapter 2 in Book V walks you through using the Add Connection dialog box.)

Adding database objects to the `DataSet` is a matter of dragging them from Server Explorer and dropping them on the design surface. Follow these steps to add new tables to DataSet Designer:

1. **Click the plus sign (+) next to the data connection you created in the preceding set of steps.**

   The data connection expands.

   Server Explorer displays data connections as a hierarchical display of database objects. You view an object's contents by clicking the plus sign (+) to expand the container.

2. **Click the plus sign (+) next to the Tables folder.**

   A list of the data connection's tables appears.

3. **Drag the Department table to the design surface and drop the table.**

   A representation of the table and a `TableAdapter` appear on the DataSet Designer. The `TableAdapter` is responsible for creating commands to retrieve and update the data for the table with which it's associated.

4. **Drag and drop the EmployeeDepartmentHistory table onto the DataSet Designer.**

   The table and a `TableAdapter` appear on the designer. Visual Studio creates links between the tables, as shown in Figure 3-7.

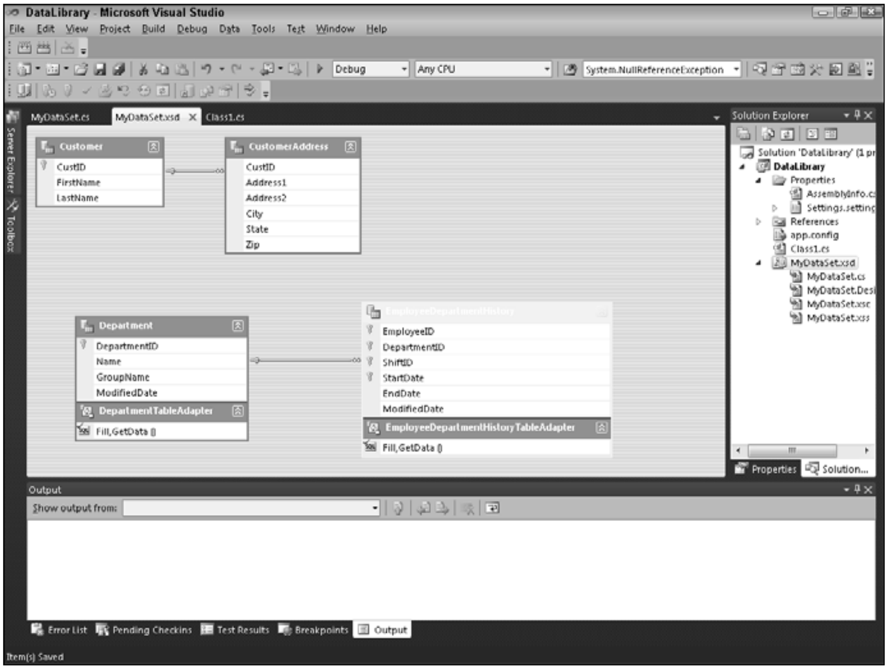


Figure 3-7: The database tables appear on the designer with links between them.

5. **Drag and drop the Employee table onto DataSet Designer.**

   The designer creates a link between the Employee table and the EmployeeDepartmentHistory table.

6. **Press Ctrl+S to save the `DataSet`.**

Notice the link at the top of the Employee table that points back to itself, as shown in Figure 3-8. The Employee table has a record for each employee. Employees have a manager, but managers are employees, too. Instead of creating a separate Manager table that duplicates information in the Employee table, the Employee table has a ManagerID column. The source of the ManagerID column is the Employee table, which creates a link. A column in a table using that table as its source is an example of a reflexive relationship.

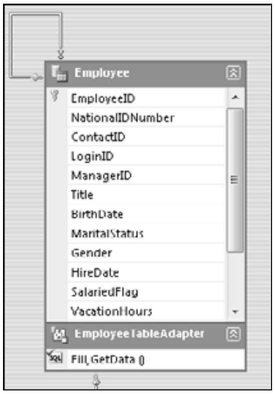


Figure 3-8: The Employee table has a link to itself.

Going beyond database tables

Tables aren't the only database objects you can add to your strongly typed `DataSet`. You can also add other kinds of database objects, such as stored procedures and views.

You access views and stored procedures by using Server Explorer. See the preceding section, "Adding database objects," for an example of using Server Explorer to access database objects.

To add a view from the Adventure Works sample database to your `DataSet`, follow these steps:

1. **Drag and drop the vEmployeeDepartment view from Server Explorer onto DataSet Designer.**

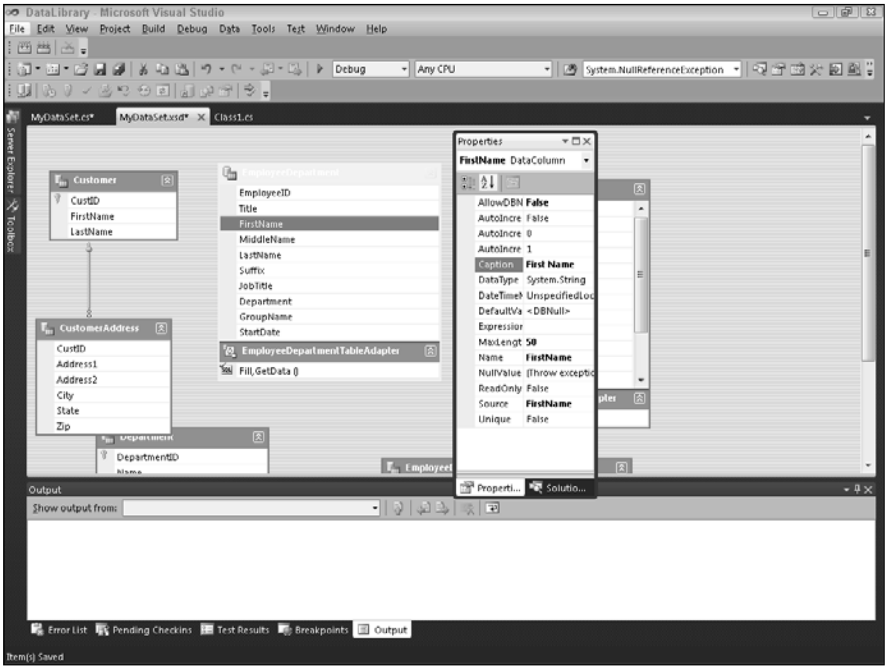
   The view appears on DataSet Designer. Views often start with the letter *v*.

2. **Click the vEmployeeDepartment DataTable name and change it to EmployeeDepartment.**
3. **Right-click the FirstName column.**
4. **Choose Properties from the contextual menu that appears.**

   The Properties window appears.

5. **Set the FirstName column's `Caption` property to FirstName, as shown in Figure 3-9.**

   Set the `Caption` property for a column to specify the text you want displayed in the column's header.



Properties

FirstName DataColumn

AllowDBN	False
AutoIncre	False
AutoIncre	0
AutoIncre	1
Caption	First Name
DataType	System.String
DateTimeM	UnspecifiedLoc
DefaultVa	<DBNull>
Expressior	
MaxLengt	50
Name	FirstName
NullValue	(Throw exceptic
ReadOnly	False
Source	FirstName
Unique	False

Figure 3-9: Set the column `Caption` property via the Properties window.

DataSet Designer can't generate SQL commands to update your data when you use a view or stored procedure. You have to use the TableAdapter Configuration Wizard to write your own SQL statements or select stored procedures. (See the next section for more information about the wizard.)

Shepherding Data

You need some way to move data in and out of your `DataSet`. The new Microsoft .NET Framework technology, the `TableAdapter`, shepherds data between your data source and `DataSet`. Each table in your `DataSet` requires its own `TableAdapter`. The `TableAdapter`, located between your database and `DataSet`, performs these functions:

- Connects to your database
- Sends SQL statements to the database to retrieve and update data

When you add a database object from Server Explorer to a `DataSet` by using DataSet Designer, the designer automatically creates a `TableAdapter`. The `TableAdapter` connects to the data source and generates an SQL statement to retrieve data to fill the table in the `DataSet`.

A single table in a `DataSet` doesn't have to correlate directly to a database table. A table in a `DataSet` can be filled with data from a view, a stored procedure, or an SQL statement you specify. See the preceding section, about going beyond database tables, for more information on adding views and stored procedures to a `DataSet`.

The DataSet Designer provides two wizards for creating `TableAdapters`:

- **TableAdapter Configuration Wizard:** Walks you through connecting to a database and creating SQL statements or stored procedures for accessing data
- **TableAdapter Query Configuration Wizard:** Allows you to add queries for accessing data

The TableAdapter wizards generate the code to make database connections and execute SQL commands by using ADO.NET to access the database. To read more about connections and SQL commands in ADO.NET, see Chapter 5 in Book V.

The TableAdapter wizards can generate SQL statements to retrieve and update data when the table's data source is a single table in a database.

If you use multiple tables in a database to fill a single table in a `DataSet`, you need to supply your own SQL statements or stored procedures to update the original data source.

Using the TableAdapter Configuration Wizard

`TableAdapter`, the workhorse of the `DataSet`, is responsible for shepherding your data in and out of the database. DataSet Designer configures a `TableAdapter` for each table you drop on the DataSet Designer.

You use the TableAdapter Configuration Wizard to

- Generate stored procedures for a single table.
- Specify the exact SQL statements you want the wizard to use.
- Bind the `TableAdapter` to existing stored procedures.

Common practice in data access is to use stored procedures to retrieve and update data. A *stored procedure* is a program that's saved with your database and is normally written by using T-SQL. You can use SQL statements to write stored procedures to do anything you want to do to your data.

For example, you can write a stored procedure to retrieve data from your database. The SQL statement to retrieve an employee looks like this:

```
SELECT EmployeeID,FirstName, LastName
FROM Employee
WHERE EmployeeID = @EmployeeID
```

Rather than execute the preceding SQL statement every time you need to get an employee's data, you create a stored procedure named `GetEmployee`, which has the SQL statement inside it. To execute the `GetEmployee` stored procedure, you type

```
EXEC GetEmployee @EmployeeID = 4
```

See the sidebar "The anatomy of a stored procedure" for more information on how to write a stored procedure.

You should use stored procedures over SQL statements because stored procedures are

- Stored centrally on the database server
- More secure than using SQL statements
- Reusable
- Easier to execute

The most commonly used stored procedures in data access are CRUD (Create, Read, Update, and Delete) procedures. The four letters in CRUD relate to each of the four SQL commands you use to retrieve and update data:

- `INSERT`

  Although the *C* in *CRUD* stands for create, the SQL command is `INSERT`. (After all, CRUD is easier to remember than ISUD.)
- `SELECT`
- `UPDATE`
- `DELETE`

Generating stored procedures

You can use the TableAdapter Configuration Wizard to create new stored procedures for a single table. By default, the wizard creates SQL statements and uses them to access data.

Stored procedures are preferred because they provide a performance advantage over regular SQL statements.

The anatomy of a stored procedure

A *stored procedure* is a set of saved SQL statements. Stored procedures can

- Accept input parameters and return values as output parameters.
- Return a status value to indicate whether the procedure succeeded or failed.

When you create a new stored procedure, you have to do a few things, such as

- Use the `CREATE PROCEDURE` statement.
- Specify parameters.
- Use the `SET` statement to send information to the database about how you want the query executed.
- Type SQL statements that operate on data, such as `SELECT`, `INSERT`, `UPDATE`, or `DELETE`.

For example, to create a new stored procedure named `GetEmployees` with one input parameter, type the following lines:

```
CREATE PROCEDURE GetEmployee
   @EmployeeID int = 0
AS
BEGIN
   SET NOCOUNT ON;
   SELECT EmployeeID,FirstName, LastName
   FROM Employee
   WHERE EmployeeID = @EmployeeID
END
GO
```

After executing this block of SQL statements, SQL Server adds the stored procedure to the database. SQL Server changes the `CREATE PROCEDURE` statement to `ALTER PROCEDURE` after the procedure is created. The remaining lines in the procedure stay the same unless you modify them.

To generate new stored procedures for the Department table, follow these steps:

1. **Right-click the Department table in DataSet Designer.**
2. **Choose Configure from the contextual menu.**

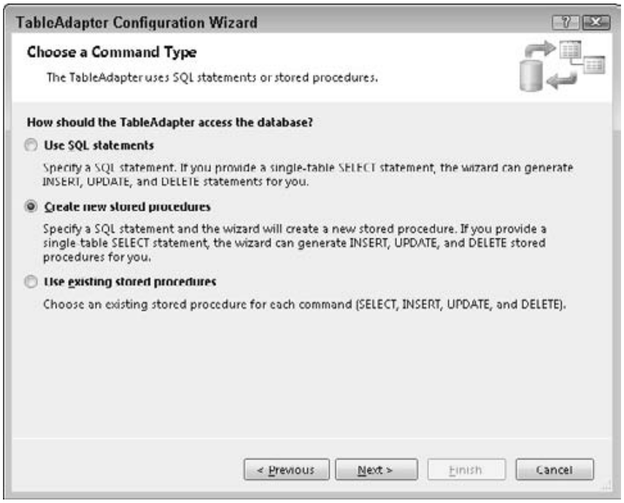
   The TableAdapter Configuration Wizard opens with the Enter an SQL Statement page displayed.

   Alternatively, you can choose Data➪Configure.

3. **Click the Previous button to step back to the preceding page in the wizard.**

   The Choose a Command Type page appears.

4. **Select the Create New Stored Procedures radio button, as shown in Figure 3-10.**



TableAdapter Configuration Wizard

Choose a Command Type

The TableAdapter uses SQL statements or stored procedures.

How should the TableAdapter access the database?

○ **Use SQL statements**

Specify a SQL statement. If you provide a single-table SELECT statement, the wizard can generate INSERT, UPDATE, and DELETE statements for you.

● **Create new stored procedures**

Specify a SQL statement and the wizard will create a new stored procedure. If you provide a single-table SELECT statement, the wizard can generate INSERT, UPDATE, and DELETE stored procedures for you.

○ **Use existing stored procedures**

Choose an existing stored procedure for each command (SELECT, INSERT, UPDATE, and DELETE).

Figure 3-10: Select the options to create new stored procedures.

5. **Click Next twice to advance to the Create the Stored Procedures page.**

6. **Type a name for each stored procedure in the appropriate text box, as shown in Figure 3-11.**

   See the sidebar “Naming stored procedures” for more information on naming stored procedures.

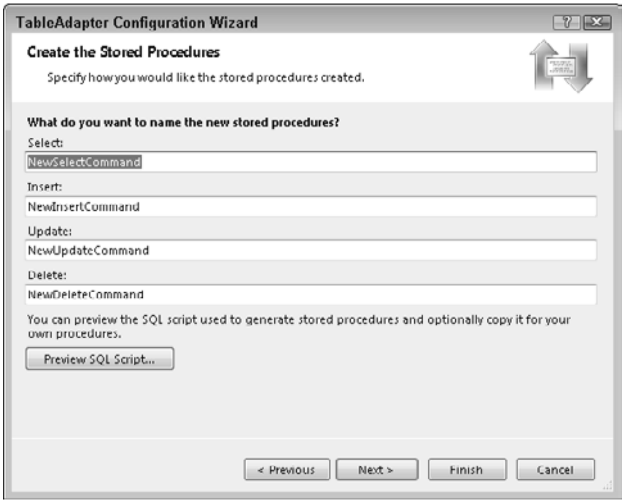
7. **Click the Preview SQL Script button to view the SQL script that Visual Studio uses to create the stored procedures.**

   You can save the script and modify it to use with other tables.

8. **Click Finish.**

   Visual Studio creates the four stored procedures in the database.

`TableAdapter` uses the new stored procedures, rather than regular SQL statements, to access your data.



TableAdapter Configuration Wizard

Create the Stored Procedures

Specify how you would like the stored procedures created.

What do you want to name the new stored procedures?

Select: NewSelectCommand

Insert: NewInsertCommand

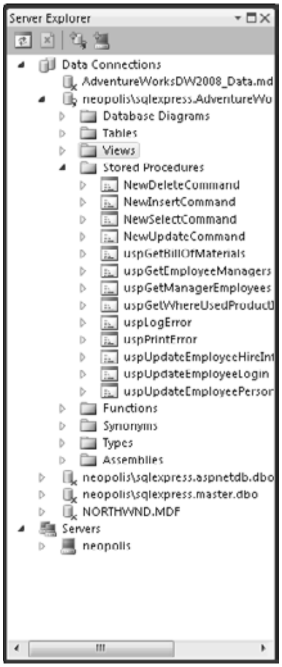
Update: NewUpdateCommand

Delete: NewDeleteCommand

You can preview the SQL script used to generate stored procedures and optionally copy it for your own procedures.

Figure 3-11: Type a name for each stored procedure.

Expand the Stored Procedures folder in Server Explorer to view the new stored procedures, as shown in Figure 3-12.



Server Explorer

```
Data Connections
    AdventureWorksDW2008_Data.md
    neopolis\sqlexpress.AdventureWo
        Database Diagrams
        Tables
        Views
        Stored Procedures
            NewDeleteCommand
            NewInsertCommand
            NewSelectCommand
            NewUpdateCommand
            uspGetBillOfMaterials
            uspGetEmployeeManagers
            uspGetManagerEmployees
            uspGetWhereUsedProductI
            uspLogError
            uspPrintError
            uspUpdateEmployeeHireInf
            uspUpdateEmployeeLogin
            uspUpdateEmployeePersor
        Functions
        Synonyms
        Types
        Assemblies
    neopolis\sqlexpress.aspnetdb.dbo
    neopolis\sqlexpress.master.dbo
    NORTHWND.MDF
Servers
    neopolis
```

Figure 3-12: View the new stored procedures in Server Explorer.

Naming stored procedures

You need a standard system for naming database objects, such as stored procedures. Most developers try to create a system that groups like procedures by using a combination of prefixes, table names, SQL commands, and clauses, such as

- `prefixTableNameCommand`
- `prefixCommandTableName`
- `prefixTableNameCommandWhere`
- `prefixCommandTableNameWhereSortby`

You can argue for and against each standard. When you're dreaming up your standard, remember that your database can easily contain thousands of stored procedures. Chances are that your scheme will break down as the quantity of stored procedures increases.

Changing your naming standard after the fact is almost impossible, so pick something and stay consistent. Naming conventions may not help you actually find something. No matter what you dream up, you'll struggle to find one stored procedure in a sea of stored procedures. Instead, your naming conventions should take the guesswork out of naming. It's one less issue you have to think about. When you're creating a stored procedure or using an existing one, the naming standard helps you know what to expect.

When you want to find stored procedures, get into the habit of using the SQL Server Management Studio to view table dependencies. (See Chapter 6 in Book V.)

Whatever your standard system for naming stored procedures, never use the prefix `sp` (short for *system procedure*). Using that prefix on your own stored procedures creates a slight performance hit because SQL Server expects `sp`-prefixed stored procedures to reside in the master database. Microsoft uses the `usp` prefix (short for *user stored procedure*), in its Adventure Works database.

Binding to existing stored procedures

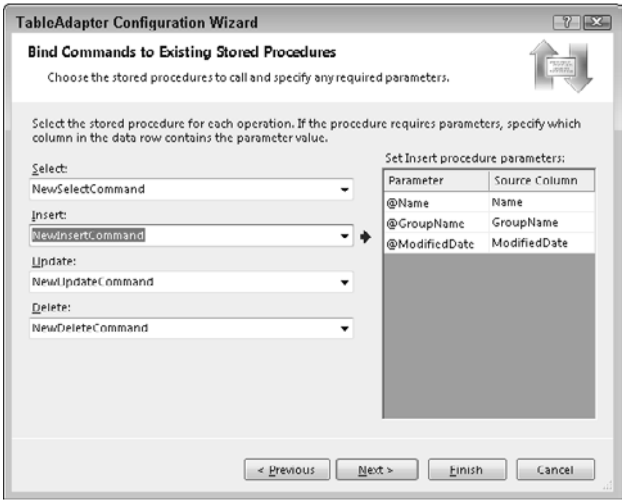
You can tell the DataSet Designer to use an existing stored procedure to access data, usually whenever you already have stored procedures created. You're likely to use this feature often because `DataSets` rarely are simple enough to use only one table.

Follow these steps to bind a `TableAdapter` to an existing stored procedure:

1. **Follow Steps 1 through 3 in the preceding section, "Generating stored procedures," to open the TableAdapter Configuration Wizard.**
2. **Select the Use Existing Stored Procedure option on the Choose a Command Type page.**
3. **Click Next.**

   The Bind Commands to Existing Stored Procedures page appears.

4. **Click the drop-down arrow for each of the commands and select the stored procedure to use, as shown in Figure 3-13.**



TableAdapter Configuration Wizard

Bind Commands to Existing Stored Procedures

Choose the stored procedures to call and specify any required parameters.

Select the stored procedure for each operation. If the procedure requires parameters, specify which column in the data row contains the parameter value.

Select: NewSelectCommand

Insert: NewInsertCommand

Update: NewUpdateCommand

Delete: NewDeleteCommand

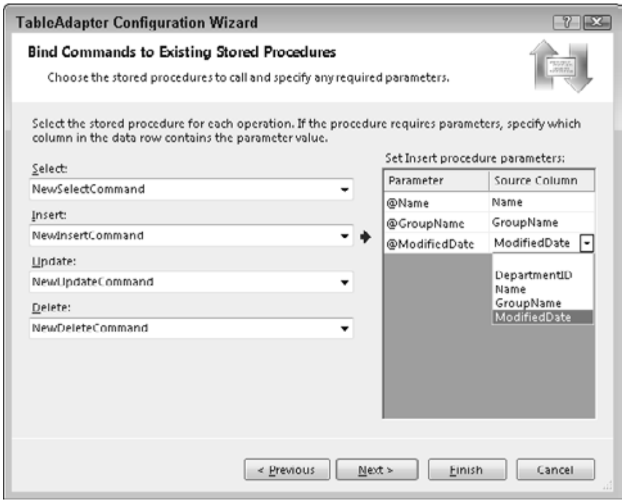
Set Insert procedure parameters:

Parameter	Source Column
@Name	Name
@GroupName	GroupName
@ModifiedDate	ModifiedDate

Figure 3-13: Select existing stored procedures from the drop-down list for each command.

5. **Verify that the source column matches the stored procedure's input parameters.**

   DataSet Designer displays the stored procedure's input parameters and their corresponding source columns from the `DataSet` table. To select a different source column, click the Source Column drop-down list and select a different column, as shown in Figure 3-14.



TableAdapter Configuration Wizard

Bind Commands to Existing Stored Procedures

Choose the stored procedures to call and specify any required parameters.

Select the stored procedure for each operation. If the procedure requires parameters, specify which column in the data row contains the parameter value.

Select: NewSelectCommand

Insert: NewInsertCommand

Update: NewUpdateCommand

Delete: NewDeleteCommand

Set Insert procedure parameters:

Parameter	Source Column
@Name	Name
@GroupName	GroupName
@ModifiedDate	ModifiedDate

Figure 3-14: Select a different source column to use as the stored procedure's input parameter.

6. **Click Finish.**

   Visual Studio generates the code to use the stored procedures.

Using the TableAdapter Query Configuration Wizard

You aren't stuck with the select statements that DataSet Designer creates for you. You can specify additional statements for retrieving and updating data.

You use the TableAdapter Query Wizard to create additional queries. To start the wizard, follow these steps:

1. **Right-click the table in the DataSet Designer.**
2. **Choose Add from the shortcut menu.**

   A submenu appears.
3. **Click Query.**

   The TableAdapter Query Configuration Wizard appears.

   Alternatively, you can access the Add submenu from the Data menu.

You can use the TableAdapter Query Wizard to create CRUD statements in the following ways:

- **Use SQL statements to access data.** The wizard automatically generates the SQL statements for single-table data sources. Otherwise, you can use the Query Builder to build your statement.
- **Create new stored procedures.** For single-table data sources, the DataSet Designer generates the SQL statements for you. You can copy and paste your SQL statement into the wizard or use the Query Builder to build the SQL statement.
- **Use existing stored procedures.** The wizard displays input parameters and results columns for the stored procedure you select.

To create an additional query to sort your output, follow these steps:

1. **Open the TableAdapter Query Wizard by following the preceding set of steps in this section.**
2. **Accept the default to use SQL statements.**

   Or, you can choose the option to create new stored procedures. Either way, DataSet Designer generates the SQL statements for you.
3. **Click Next.**

   The Choose a Query Type page appears.
4. **Accept the `SELECT` default type, which returns rows.**

5. **Click Next.**

   The Specify an SQL SELECT statement page appears.

   DataSet Designer generates the following SQL statement, which appears in the wizard:

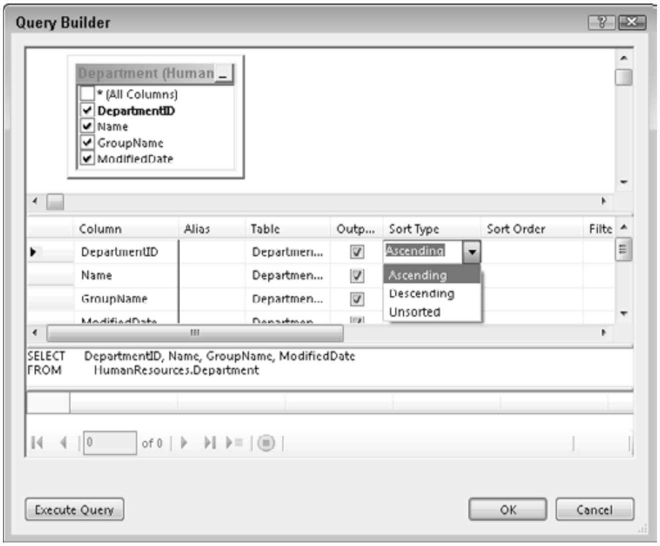
   ```
   SELECT DepartmentID, Name, GroupName, ModifiedDate FROM HumanResources.
       Department
   ```

6. **Click the Query Builder button.**

   The Query Builder appears.

7. **Click the Sort Type drop-down list on the column grid.**

8. **Select Ascending from the Sort Type drop-down list, as shown in Figure 3-15.**



☐ * (All Columns)
☑ DepartmentID
☑ Name
☑ GroupName
☑ ModifiedDate

	Column	Alias	Table	Outp...	Sort Type	Sort Order	Filte
▸	DepartmentID		Departmen...	☑	Ascending		
	Name		Departmen...	☑			
	GroupName		Departmen...	☑			
	ModifiedDate		Departmen...	☑			

```
SELECT   DepartmentID, Name, GroupName, ModifiedDate
FROM       HumanResources.Department
```

Figure 3-15: Select Ascending from the Sort Type drop-down list in the column grid.

9. **Press Enter.**

   The Query Builder adds the following clause to the SQL statement:

   ```
   ORDER BY DepartmentID
   ```

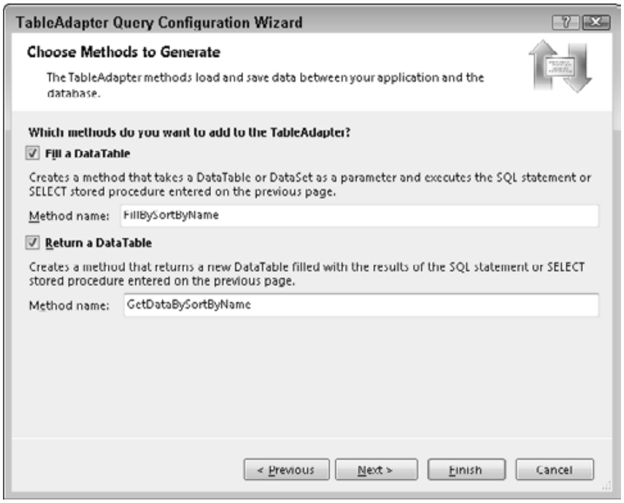
10. **Click OK.**

    The updated SQL statement appears in the wizard.

11. **Click Next.**

    The Choose Methods to Generate page appears.

12. **Type names that you want the wizard to use in generating methods, as shown in Figure 3-16.**



TableAdapter Query Configuration Wizard

Choose Methods to Generate

The TableAdapter methods load and save data between your application and the database.

Which methods do you want to add to the TableAdapter?

☑ Fill a DataTable

Creates a method that takes a DataTable or DataSet as a parameter and executes the SQL statement or SELECT stored procedure entered on the previous page.

Method name: FillBySortByName

☑ Return a DataTable

Creates a method that returns a new DataTable filled with the results of the SQL statement or SELECT stored procedure entered on the previous page.

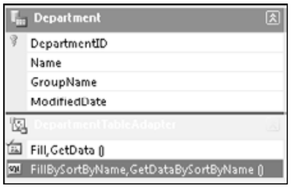
Method name: GetDataBySortByName

Figure 3-16: Type names you want the wizard to use in generating methods.

- *For the Fill a DataTable method,* type **FillBySortByName**.
- *For the Return a DataTable method,* type **GetDataBySortByName**.

13. **Click Finish.**

    DataSet Designer generates the code using the SQL statement you enter in the wizard. The designer updates the `TableAdapter` to show the new query, as shown in Figure 3-17.



Department

DepartmentID
Name
GroupName
ModifiedDate

DepartmentTableAdapter

Fill,GetData ()
FillBySortByName,GetDataBySortByName ()

Figure 3-17: The `Table-Adapter` displays the new query.

Looking under the Hood of a Strongly Typed DataSet

DataSet Designer generates an XML schema and some ADO.NET code while you're creating a strongly typed `DataSet`. Creating a `DataSet` with only a few tables can generate thousands of lines of code. You may want to view that code so that you can understand how to build XML schemas or work with ADO.NET.

Viewing code

When you drop database objects on the design surface, Visual Studio generates code "behind the scenes." To view the code that Visual Studio generates, follow these steps:

1. **Click the Show All Files button on the Solution Explorer toolbar.**
2. **Click the plus sign (+) next to your XSD file.**
3. **Double-click the file with the extension `.Designer.cs file`.**

   Visual Studio opens the file in the Code Editor.

Don't change the code generated by Visual Studio. You can't be sure that the program won't overwrite your changes.

Using partial classes

You can add your own code to extend the typed `DataSet`. The typed `DataSet` takes advantage of partial classes. A *class* is a unit of code. In previous versions of .NET, all the code for a class had to be in a single file. Partial classes can span multiple files. As long as each partial class shares the same name and uses the keyword `Partial`, Visual Studio smooshes them together when it compiles the class.

Partial classes are intended to separate Visual Studio–generated code from the code you write. As a rule, you don't need to create new partial classes.

For example, you can use a partial class to set the default value of a column or to set validation. Follow these steps to set a column's default value in a partial class for the Department table from the earlier section "Adding database objects":

1. **Open the typed `DataSet` you can read how to create in the earlier section "Adding database objects".**

   The typed `DataSet` contains a partial class generated by Visual Studio called `DepartmentDataTable`.

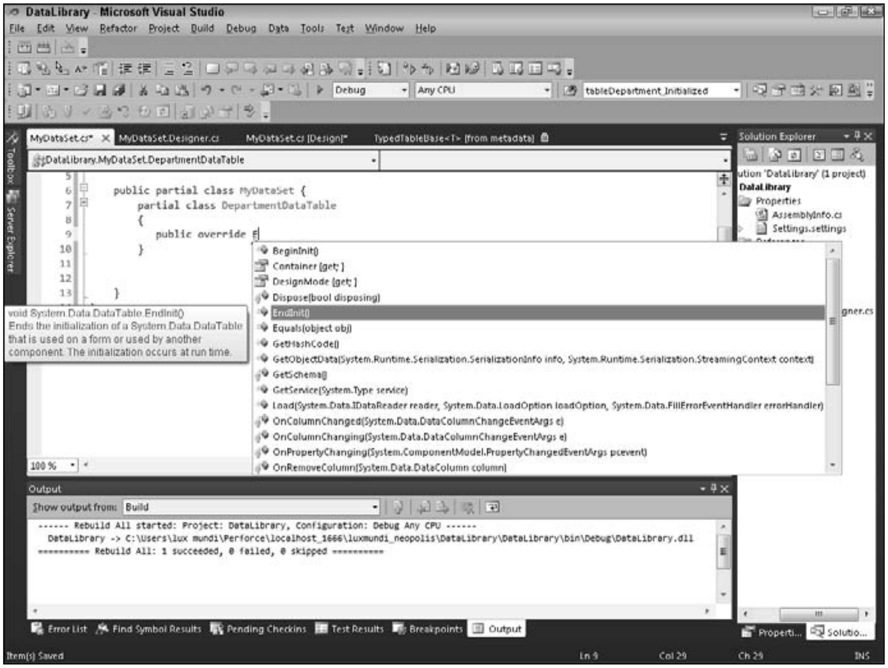
When Visual Studio generates code for typed `DataSet`s, it names the class by appending `DataTable` to the end of the table's name.

2. **Right-click the DataSet Designer.**
3. **Choose View Code from the contextual menu.**

   The Code Editor appears.
4. **Position your cursor between the start and end braces for the `Partial` class in the Code Editor.**
5. **Type the following line:**

   ```
   partial class DepartmentDataTable
   ```

   As you type, IntelliSense lists the available partial classes from the generated typed `DataSet` code. IntelliSense is a feature of the Code Editor that helps you write code.
6. **Select or type** DepartmentDataTable**.**
7. **Press Enter.**
8. **Select DepartmentDataTable from the Class drop-down list in the upper-left corner of the Code Editor, as shown in Figure 3-18.**



```
    public partial class MyDataSet {
        partial class DepartmentDataTable
        {
            public override E
        }


    }
```

BeginInit()
Container [get;]
DesignMode [get;]
Dispose(bool disposing)
EndInit()
Equals(object obj)
GetHashCode()
GetObjectData(System.Runtime.Serialization.SerializationInfo info, System.Runtime.Serialization.StreamingContext context)
GetSchema()
GetService(System.Type service)
Load(System.Data.IDataReader reader, System.Data.LoadOption loadOption, System.Data.FillErrorEventHandler errorHandler)
OnColumnChanged(System.Data.DataColumnChangeEventArgs e)
OnColumnChanging(System.Data.DataColumnChangeEventArgs e)
OnPropertyChanging(System.ComponentModel.PropertyChangedEventArgs pcevent)
OnRemoveColumn(System.Data.DataColumn column)

void System.Data.DataTable.EndInit()
Ends the initialization of a System.Data.DataTable that is used on a form or used by another component. The initialization occurs at run time.

```
------ Rebuild All started: Project: DataLibrary, Configuration: Debug Any CPU ------
  DataLibrary -> C:\Users\lux mundi\Perforce\localhost_1666\luxmundi_neopolis\DataLibrary\DataLibrary\bin\Debug\DataLibrary.dll
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

Figure 3-18: Use IntelliSense to implement the `EndInit` method.

9. **Choose `EndInit` from the IntelliSense drop-down list.**

   Visual Studio inserts the following implementation for the `EndInit` in `DepartmentDataTable`:

   ```
   public override void EndInit()
   {
      base.EndInit();

   }
   ```

   The `EndInit` is called after the data table completes its initialization. Visual Studio generates the code for initializing the data table, and this code is responsible for creating all the columns in the table.

10. **Type the following line to set the ModifiedDate column's default value to the current date and time:**

    ```
    this.columnModifiedDate.DefaultValue = System.DateTime.Now;
    ```

    The ModifiedDate column displays the current date and time when a new row is created.

Updating a ModifiedDate column when rows are inserted or updated in a table is a common practice. You can return the `ModifiedDate` value in the rows to test for optimistic concurrency violations, which assumes you won't have any data concurrency issues. With optimistic concurrency, you don't lock rows when they're read from a table. `DataSet`s use optimistic concurrency.

Using the debugger to step through the code generated by Visual Studio is a valuable way to learn new coding techniques. See Book III, Chapter 7 for more information on debugging.

Viewing the source

You can view the XML Schema of a `DataSet`. To view the XML schema of a typed `DataSet`, follow these steps:

1. **Right-click a `DataSet` in Solution Explorer.**

   For more information on adding `DataSets` to a project, see the section "Creating Typed DataSets," earlier in this chapter.

2. **Choose Open With from the contextual menu.**
3. **In the Open With dialog box, select the editor you want to use, as shown in Figure 3-19.**

   - *Open the `DataSet` in a visual XML schema designer.*
   - *Choose XML Editor to view the XML source for the `DataSet`.*

4. **Click OK.**

   The schema opens in the editor of your choice.

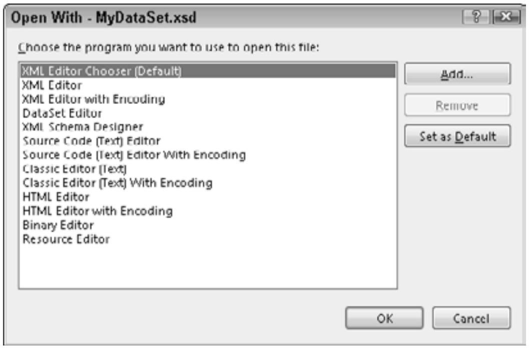

Open With - MyDataSet.xsd
Choose the program you want to use to open this file:
XML Editor Chooser (Default)
XML Editor
XML Editor with Encoding
DataSet Editor
XML Schema Designer
Source Code (Text) Editor
Source Code (Text) Editor With Encoding
Classic Editor (Text)
Classic Editor (Text) With Encoding
HTML Editor
HTML Editor with Encoding
Binary Editor
Resource Editor
Add...
Remove
Set as Default
OK
Cancel

Figure 3-19: Choose the editor you want to use to view the DataSet schema.

You can open any XML schema file with DataSet Designer. In the preceding steps, substitute your XML schema file in Step 1 and choose the DataSet Designer in Step 3. To read more about creating XML schemas, see Chapter 4 in Book V.

DataSet Designer marks up the standard syntax of an XML schema so that the schema can be used to generate ADO.NET code. By default, the attribute designating that an XML schema should open in the DataSet Designer is

```
msdata:IsDataSet="true"
```

Using a Typed DataSet

Merely creating a `DataSet` isn't enough. You must configure your application to use the `DataSet` in some way. Earlier in this chapter, you can read how to create a typed `DataSet` in a class library (in the section "Creating Typed DataSets"). This section, however, shows you how to use a class library in a Windows project.

Typed `DataSets` are reusable. After you create a `DataSet` for one application, you can reuse the `DataSet` in another project. You might think that this refers to copying the `DataSet` file from one project to another. Although the copy-and-paste method is one means of "reusing," a more elegant approach is to create your `DataSet` in a class library, as described in the earlier section "Creating a DataSet in a class library," and use that library in other projects.

You can use any .NET programming language you wish when you create your class library in Visual Studio. The programming language of the class library and the project that uses the class library can be different.

To use a class library in a Windows Form, follow these steps:

1. **Open or create your Windows project and Form wherever you want to display the `DataSet`.**
2. **Choose Data➪Add New Data Source.**

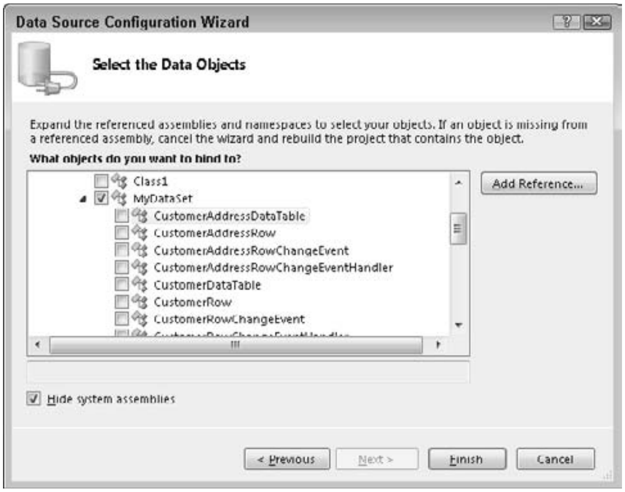
   The Data Source Configuration Wizard appears.
3. **Click the Object icon on the Choose a Data Source Type page.**
4. **Click Next.**
5. **On the Select the Object You Wish to Bind to page, click the Add Reference button.**

   The Add Reference dialog box appears.

   By adding a reference to the class library in your Windows project, you can access the class library.
6. **Click the Browse tab and browse to the folder where your class library is saved.**
7. **Select the DLL file that was generated when you built your class file.**

   The class library appears in the wizard, and the DLL file is stored in the project's Bin folder.
8. **Select MyDataSet, as shown in Figure 3-20.**
9. **Click Finish.**

   The wizard adds the class library as a data source.



Data Source Configuration Wizard

Select the Data Objects

Expand the referenced assemblies and namespaces to select your objects. If an object is missing from a referenced assembly, cancel the wizard and rebuild the project that contains the object.

What objects do you want to bind to?

- □ Class1
- ☑ MyDataSet
  - □ CustomerAddressDataTable
  - □ CustomerAddressRow
  - □ CustomerAddressRowChangeEvent
  - □ CustomerAddressRowChangeEventHandler
  - □ CustomerDataTable
  - □ CustomerRow
  - □ CustomerRowChangeEvent

☑ Hide system assemblies

Figure 3-20: Select the `DataSet` after adding a reference to the class library.

To use the data source with a databound control, follow these steps:

1. **Drag and drop the table from the `DataSet` in the Data Sources window onto the Windows Form.**

   The Windows Form Designer adds a `DataGridView` control that's bound to the `DataSet` on the form.

2. **Double-click the Windows Form.**

   The designer creates an event handler for the form's `Load` event and displays the handler in the Code Editor.

3. **Below the event handler, create a new variable to hold the table's `TableAdapter` by typing the following code:**

   ```
   private DataLibrary.MyDataSetTableAdapters.DepartmentTableAdapter
       myTableAdapter;
   ```

   When you add the `DataSet` to the form, the Windows Form Designer generates the code to display the `DataSet` in the `DataGridView` control. Unfortunately, because the designer doesn't wire up the table adapter to automatically load data, the `DataSet` doesn't have any data.

4. **In the form's `Load` event handler, type the following lines:**

   ```
   this.myTableAdapter = new DataLibrary.MyDataSetTableAdapters.
       DepartmentTableAdapter();

               this.myTableAdapter.Fill(this.myDataSet.Department);
   ```

   This code uses the `TableAdapter` you create in Step 3 to fill the `DataSet`.

Follow these steps to enable `DataGridView` to edit and update data:

1. **On the Form, click the Save Item button on the `BindingNavigator` component.**
2. **Press F4 to display the button's properties.**
3. **Set the button's `Enabled` property to true.**
4. **Double-click the Save Item button.**

   The Windows Form Designer creates an event handler for the Save Item button `Click` event.

5. **Type the following code in the Save Item button `Click` event:**

   ```
   this.Validate();
   this.departmentBindingSource.EndEdit();
   this.myTableAdapter.Update(this.myDataSet.Department);
   ```

6. **Press Ctrl+F5 to run your Windows application.**

   The Windows Form displays the data from the `DataSet` in the `DataGridView` control. As you use `DataGridView` to modify the data, the updates are saved in the `DataSet`.

7. **Click the Save Item button to save the data to the database.**

TIP

Use the `ObjectDataSource` control in ASP.NET to connect a typed `DataSet` to databound controls in a Web site. See Chapter 2 in Book V for a walkthrough of using the `ObjectDataSource` control.

Chapter 4: Working with XML

In This Chapter

- **Creating and storing XML files in Visual Studio**
- **Using XML Designer to create an XML Schema**
- **Transforming XML with style sheets**
- **Using XML with the .NET Framework**

With Visual Studio, you can use XML without writing any code whatsoever. This chapter shows you how to use the tools that Visual Studio provides for generating XML, including

- XML Editor with IntelliSense and validation
- XSLT Debugger
- XML Schema Designer
- XML snippets

Exploring XML

XML is the gold standard for working with data in modern software application development. XML, short for *eXtensible Markup Language,* is a markup language like HTML, which is the language used to create Web pages. In HTML, you use a set of predefined markup tags, such as `<html>` and `<body>`. The tags use attributes to further define the markup, such as `<body background="white">`.

XML uses tags and attributes like HTML does. However, unlike in HTML, you define what tags and attributes to use. XML is used to describe data, so you make up your own tags and attributes to define the data. For example, an XML file could use these tags to describe data about a book collection:

```
<Books>
<Book>
<Author></Author>
<Title></Title>
<Book>
</Books>
```

The XML file *might* use these tags because no predefined rules specify the tags you use. You decide which ones to use. As long as you declare which tags you want to use and then use them consistently, anyone can understand how to read your XML file. As you can see, XML is a powerful language for describing data because XML files

- Are plain-text files that you can edit and read in any text editor
- Include a schema definition that identifies all the elements and attributes to use to describe data
- Can transform into other languages, such as HTML

Microsoft has wholeheartedly embraced XML, and Visual Studio has several intelligent editors for creating and manipulating XML files. The .NET Framework has an entire namespace for reading and writing XML files, and Microsoft SQL Server supports XML, as well. If you're a .NET developer, you have all the support you need for creating and consuming XML.

REMEMBER

DataSet Designer generates an XML Schema to support the creation of strongly typed `DataSet`s. (See Chapter 3 in Book V to read more about DataSet Designer.)

Having some understanding of XML technologies is helpful as you work with the Visual Studio Tools. Don't worry: You don't have to be a whiz. This chapter has everything you need to know to get started using XML in Visual Studio. You can fill in the gaps in your XML knowledge as you go along. You can find some great tutorials at

```
www.w3schools.com
```

Storing Data the XML Way

More than a decade ago, developers had few options for transmitting data between disparate systems or storing configuration data. Everybody used comma- or tab-delimited, plain-text files. Although XML is stored in a plain-text file, the power of XML lies in its ability to describe itself. Walk through the following scenario to see how XML has transformed data transmission.

Imagine that you need to send purchase order data to a supplier. Here's what the transaction might look like without XML:

1. **You export the data from your database to a plain-text file, either comma- or tab-delimited.**
2. **You prepare a document that explains each of the columns in the output file, such as the data type and whether the data is required.**
3. **You send the file and the document to your supplier.**

4. **Your supplier tells you that the file doesn't meet the specifications for his system and sends you the specifications.**
5. **You realize that you have to hire a programmer to transform your data to meet the supplier's specifications.**
6. **You decide that it's too much trouble and keep faxing your order forms.**

This scenario has played itself out in many variations over the years. Somebody gets the bright idea to submit data electronically. However, without agreed-upon standards, the idea quickly gets abandoned because too much custom programming is usually required.

This scenario uses XML instead:

1. **You create an XML Schema file that describes all the fields and attributes in your XML file.**
2. **You export your data to an XML file that conforms to the XML Schema.**
3. **You send your XML file, along with the schema, to your suppliers.**
4. **Each supplier uses the schema to validate that your XML file contains all the required fields and attributes.**
5. **Each supplier uses a style sheet that transforms your XML file into the XML file that conforms to the schema for their corporate databases.**

Here's another scenario that may hit close to home for you. You use Word or another Office application to provide the form on-screen. The user fills in the data but saves the output as XML, rather than as a DOC or a DOCX file. Even though the user sees a form, the system sees XML. You can provide some great automation for this kind of setup by using Visual Studio Tools for Office (VSTO). You can actually use this kind of setup to convert invoice letters to XML. You can print the invoice letter as a letter or send it to someone as a DOC file yet keep your personal storage as an XML file, which takes considerably less space than a DOC file and is easier to manipulate for other uses (such as accounting or statistics).

Of course, not everybody plays nice all the time. A handful of suppliers may reject your file and tell you that they don't use XML, or they might want to use a comma-delimited file. It's no big deal — just create a style sheet (known as Extensible Stylesheet Language, or XSLT) that transforms your XML file into a comma-delimited file that meets their requirements.

This example shows you the beauty of XML. You can accommodate the input and output requirements of multiple systems without having to write a new program every single time. As long as your program knows how to work with the XML, you can make all the changes you need.

Using XML to define your data is fairly straightforward, especially when you harness the power of Visual Studio. A little knowledge of XML, combined with the Visual Studio tools, goes a long way toward creating XML.

Use XML to describe data any time you need to

- Transmit data between disparate systems.
- Store configuration data for your application.
- Avoid the overhead of a relational database management system, such as SQL Server.

XML files can get really bulky when used to store large quantities of data, which doesn't make them appropriate for all data transmission scenarios. You should try to send the minimal amount of data required to complete a transaction when using XML.

Creating a New XML File

The first step in using the XML editors in Visual Studio is to create a new XML file. For example, Visual Studio supports creating these types of XML files:

- **XML document:** Stores self-describing data
- **XML Schema:** Defines the elements and attributes used in an XML document to describe data
- **XSLT style sheet:** Transforms an XML document into another format, such as XML or HTML

XML documents, XML Schemas, and XSLT style sheets — they're all XML files. XML files use XML markup tags. XML Schemas and XSLT style sheets use a set of predefined XML tags.

To create an XML file in Visual Studio, follow these steps:

1. **Choose File➪New.**
2. **Choose File from the contextual menu that appears.**

   The New File dialog box appears.

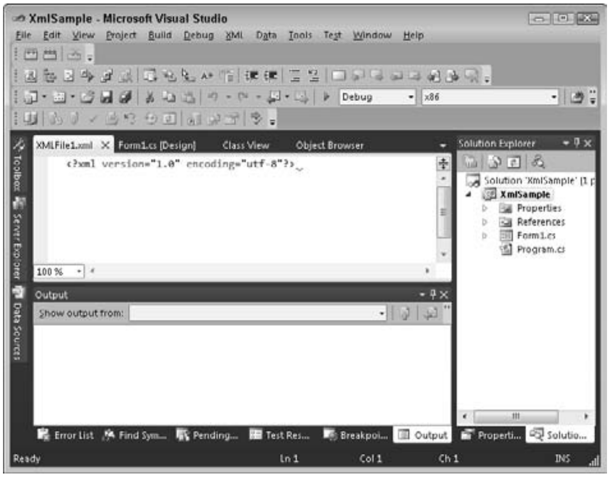
   Alternatively, you can press Ctrl+N on your keyboard to open the New File dialog box.

3. **Click the icon for the type of XML file to create: XML File, XML Schema, or XSLT File.**

   Clicking the XML File icon creates an XML document file.

4. **Click Open.**

   Visual Studio displays the XML file in the appropriate editor. Figure 4-1 shows an XML document in the XML Editor.



```
<?xml version="1.0" encoding="utf-8"?>
```

Figure 4-1: A blank XML file appears in the editor.

All XML files require a declaration on the file's first line. This declaration is a processing instruction that is important when reading the XML data. Visual Studio adds the following XML declaration for you:

```
<?xml version="1.0" encoding="utf-8"?>
```

The declaration specifies which version of XML you're using and in which language the content is encoded. You don't need to change the declaration.

Describing Data in an XML Document

Suppose that you want to create a Web site to display your large set of recipes. You can easily store those recipes in an XML document and then transform them to HTML so that they can be displayed in a browser.

For each recipe, you need to store the following information:

- The recipe name
- A list of ingredients, including quantities
- A list of step-by-step instructions

You can use the Visual Studio XML Editor to create an XML document that describes a set of recipes.

XML files contain markup tags to describe the data they store.

To create an XML document for storing data, such as the recipes in this example, follow these steps:

1. **Create a new XML document file by following the steps in the preceding section.**

2. **In a blank XML document file, type an opening root tag, such as <recipes>.**

   Visual Studio inserts the closing tag `</recipes>` for you, and then the XML Editor displays `<recipes></recipes>` in the XML document.

   XML uses opening and closing tags to describe data. The `<recipes>` tag is the XML document's root tag; in this case, it encloses a collection of recipes. A *root tag* is the outermost tag in an XML document.

3. **Between the recipes' tags, type** <recipe></recipe>**.**

   Use the `<recipe>` tags to enclose a single recipe.

4. **Inside the first `<recipe>` tag, type** name=`""`.

   The tags appear as `<recipe name=""></recipe>`.

   `name` is an attribute of recipe. XML tags use attributes similar to HTML tags. *Attributes* allow you to describe properties for the tag.

5. **Between the `<recipe>` tags, type** <ingredients>**.**

   The `<ingredients>` tag stores a collection of individual `<ingredient>` elements.

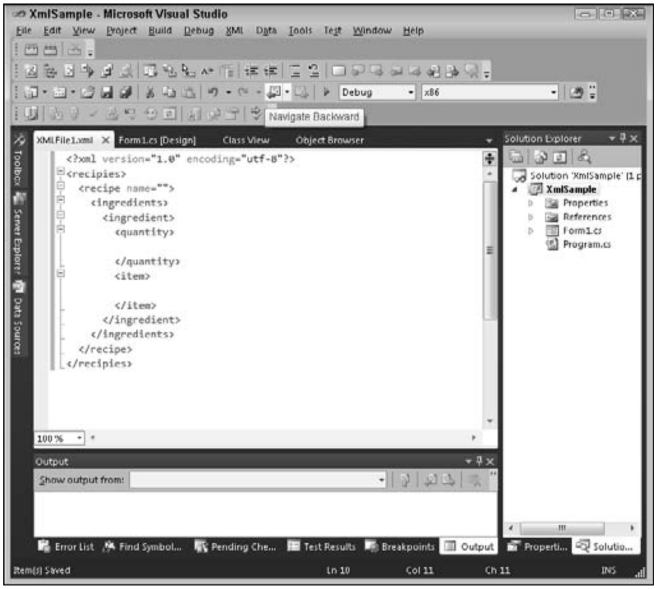
6. **Between the `<ingredients>` tags, type** <ingredient>**.**

7. **Inside the `<ingredient>` tag, add tags for quantity and item.**

   Figure 4-2 shows an example of the XML file.

   A blank XML file appears in the editor.

8. **Save your file as `recipe.xml`.**



```
<?xml version="1.0" encoding="utf-8"?>
<recipies>
  <recipe name="">
    <ingredients>
      <ingredient>
        <quantity>

        </quantity>
        <item>

        </item>
      </ingredient>
    </ingredients>
  </recipe>
</recipies>
```

Figure 4-2: Create an XML document file to describe your data.

All's well (and valid) that's formed well

When you create an XML document, you want your document to be well-formed and valid. An XML file is considered well-formed if all tags

- Are contained within a single document root tag, such as `<recipes>`
- Are properly nested so that inner tags are closed
- Are opened and closed properly
- Possess all the proper punctuation, such as quotation marks around attribute values

An XML file is valid if the document is

- Well-formed
- Consistent with the data schema defined for the document

You know whether your document is well-formed because the XML Editor displays squiggly lines under any incorrect syntax. In this chapter, you can read how to validate an XML document by using an XML Schema.

Creating an XML Schema

The value of XML lies in its self-describing documents. One technology that makes it possible for XML data to describe itself is the XML Schema. If you ever work with databases, you probably recognize the word *schema.* A *schema* describes all the tables, columns, and data types in a database. An XML Schema does the same thing.

A schema is used to verify an XML document's validity. XML files are opened by using a processor. A schema allows an XML file to respond to the processor's query, "Tell me about yourself." The processor uses the schema's response about the XML document to determine whether the XML document is valid.

An XML file must be well-formed in order to be valid. The properties of a well-formed XML file are described in the sidebar "All's well (and valid) that's formed well."

Building an XML Schema

The syntax for creating an XML Schema is straightforward to understand but somewhat tedious to write. Never fear: Visual Studio provides a tool for generating an XML Schema. Visual Studio generates the schema by inferring the schema from an existing XML document. Inferring a schema from an existing document and then tweaking it is much easier than building the schema from scratch.

Generating an XML Schema from an existing XML document is more accurate when the document contains data. Having data in the file allows Visual Studio to make intelligent guesses about how the data is structured in the file. For more information on creating XML Schemas from scratch, see the W3 Schools tutorial at

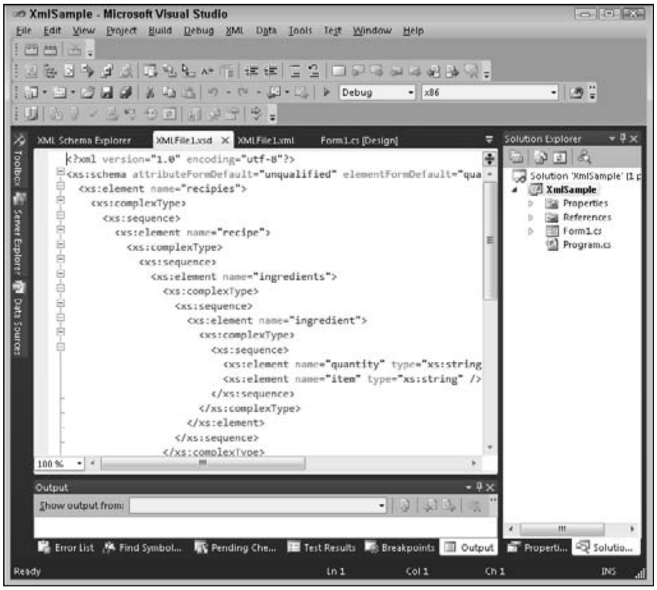
```
www.w3schools.com/schema
```

To infer the schema for an existing XML document, follow these steps:

1. **Open the XML document in Visual Studio.**
2. **Choose XML⇨Create Schema.**

   Visual Studio generates the schema file and displays it in the Code Editor.

   Figure 4-3 shows the schema that's inferred for the `recipe.xml` file you can create in the earlier section "Describing data in an XML document." See the next section for more information on understanding the XML Schema syntax.



```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qua
  <xs:element name="recipies">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="recipe">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ingredients">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ingredient">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="quantity" type="xs:string
                          <xs:element name="item" type="xs:string" />
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
```

Figure 4-3: Visual Studio infers the schema from an existing XML file.

3. **Save the schema file.**

   XML Schema files use the `.xsd` file extension.

Visual Studio associates the schema file with the XML file by using the XML document file's `Schema` property. Press F4 to display the XML document's properties.

Associating the schema with your XML document does two things:

- Validates your XML document against the XML Schema
- Allows you to use IntelliSense in the XML Editor

Using XML snippets

The XML Editor uses XML snippets to insert commonly used XML tags. In this context, a *snippet* is a block of XML tags. You can create your own snippets, or you can quickly insert a snippet if your XML document is associated with a schema.

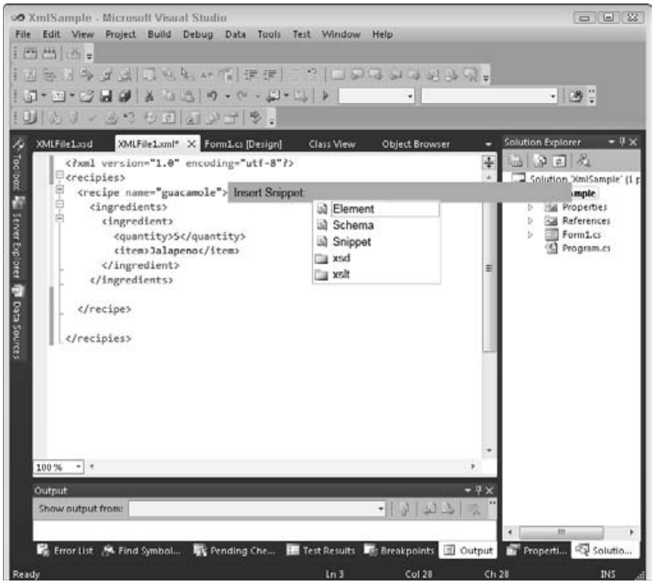
XML snippets are a kind of code snippet, which is an IntelliSense feature. You use code snippets to insert commonly used C# and Visual Basic code. See Book III, Chapter 5 for more details on using code snippets.

To use an XML snippet with the `recipe.xml` file described in the earlier section “Describing data in an XML document,” follow these steps:

1. **Position your cursor after the guacamole recipe.**
2. **Type an opening tag: <.**

   IntelliSense displays a list of available elements based on the document’s schema.
3. **Press your down-arrow key to select the recipe element from the drop-down list and then press Enter.**
4. **Press the Tab key.**

   Visual Studio inserts an XML snippet from the schema, as shown in Figure 4-4. The recipe tags and everything in between are inserted.



```
<?xml version="1.0" encoding="utf-8"?>
<recipies>
  <recipe name="guacamole">
    <ingredients>
      <ingredient>
        <quantity>5</quantity>
        <item>Jalapeno</item>
      </ingredient>
    </ingredients>

  </recipe>

</recipies>
```

Figure 4-4: Press Tab to insert an XML snippet based on the document’s schema.

The areas that need to have text entered are highlighted in the Code Editor. Use the Tab key to move between the highlighted areas.

To create your own XML snippets from scratch, follow these steps:

1. **Right-click the XML document and choose Insert Snippets from the contextual menu.**
2. **Choose Snippet from the IntelliSense menu.**

   Visual Studio inserts a boilerplate template for creating XML snippets.

3. **Replace the boilerplate content with your own XML tags.**

   See the Visual Studio help system for more information.

4. **Save your file with the filename extension `.snippet`.**

Transforming XML with Extensible Stylesheet Language Transformations

The role of XML is to store self-describing data documents. At some point, that data needs to be displayed. You use a special kind of style sheet that uses XSLT to convert XML to another language, such as HTML, that you can use to display the file's contents.

You can use XSLT to transform XML documents into

- XML documents.
- HTML documents.
- XHTML (Extensible Hypertext Makeup Language) documents.
- Plain text documents.
- Programming languages, such as C# or SQL.
- Other kinds of data documents: The uses are nearly unlimited.

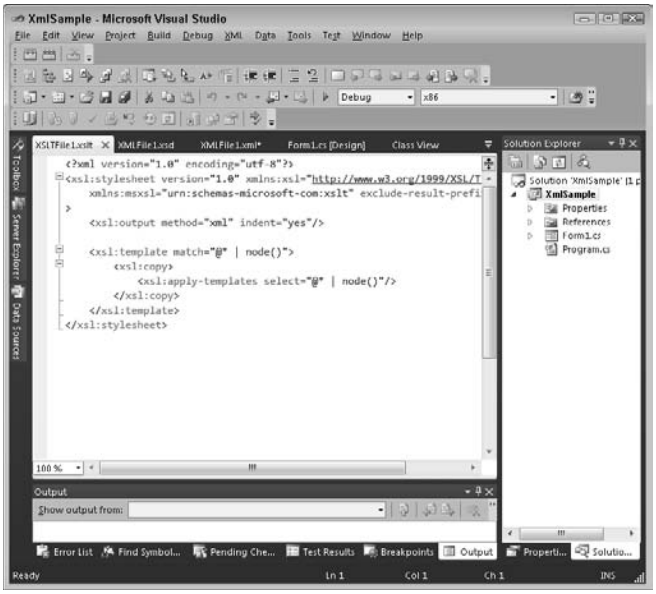
Visual Studio provides a file template and a Code Editor for creating XSLT files. Like all the Code Editors in Visual Studio, the XSLT Editor includes support for IntelliSense and code snippets.

Visual Studio can apply a style sheet to your XML file and display the output. An XSLT debugger is available to help you track down errors in your style sheet.

To create a new XSLT style sheet in Visual Studio, follow these steps:

1. **Press Ctrl+N to open the New File dialog box.**
2. **Click the XSLT File icon.**
3. **Click Open.**

   An XSLT file appears in the Code Editor, as shown in Figure 4-5.



```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/T
    xmlns:msxsl="urn:schemas-microsoft-com:xslt" exclude-result-prefi
>

    <xsl:output method="xml" indent="yes"/>

    <xsl:template match="@* | node()">
        <xsl:copy>
            <xsl:apply-templates select="@* | node()"/>
        </xsl:copy>
    </xsl:template>
</xsl:stylesheet>
```

Figure 4-5: An XSLT file appears in the Code Editor.

Unfortunately, no visual designer or code-generation tool in Visual Studio creates XSLT style sheets. Another glaring oversight is a lack of support for building the expressions you need for selecting nodes from your XML file.

The technology for selecting nodes from an XML document is XPath. You can read more about it later in this section.

If you're having trouble building expressions, use InfoPath to dynamically build XPath expressions. InfoPath 2003 provides a visual tool for building XPath expressions. You can copy and paste the expressions from InfoPath into your style sheet. You can also use XML Notepad 2007, which you can download from

```
www.microsoft.com/downloads/details.aspx?familyid=72D6AA49-787D-4118-BA5F-
    4F30FE913628
```

Writing an XSLT style sheet

XSLT is written in XML. A style sheet consists of the following elements:

- **Declaration:** Identifies the file as an XSLT style sheet
- **Templates:** Define the style sheet's output
- **Supplementary content:** HTML tags, for example, that you want to present in the output

The "meat and potatoes" of an XSLT style sheet consists of the template (or templates) that defines the style sheet's output. The following example of an XSLT template uses the `recipe.xml` file that you can read how to create earlier in this chapter. Notice that the template combines XSLT syntax with HTML:

```
<xsl:template match="/">
  <html>
    <body>
      <xsl:for-each select="recipes/recipe">
        <ul>
          <xsl:for-each select="ingredients/ingredient">
            <li>
              <xsl:value-of select="quantity" />
              <xsl:text> </xsl:text>
              <xsl:value-of select="item" />
            </li>
          </xsl:for-each>
        </ul>
      </xsl:for-each>
    </body>
  </html>
</xsl:template>
```

The output from this template looks like this:

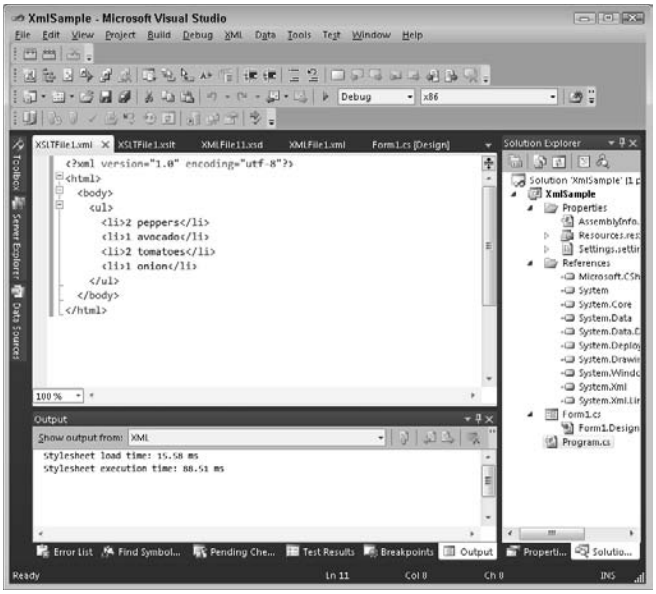
```
<html>
  <body>
    <ul>
      <li>2 peppers</li>
      <li>1 avocado</li>
      <li>2 tomatoes</li>
      <li>1 onion</li>
    </ul>
  </body>
</html>
```

The HTML tags are composed of output from the style sheet. The ingredients data is pulled from the `recipe.xml` document by using XPath expressions.

An XSLT template includes these items:

- Elements, such as `for-each` and `value-of`, that are used to operate on the data from the XML document
- XPath expressions used in the `select` attribute of a template's elements to navigate the XML document
- Functions, such as `last()` and `position()`, for evaluating whether to select a node or include text in the output
- Text, such as HTML markup tags, for inclusion in the output

Figure 4-6 shows an example of an XSLT style sheet used to transform the `recipe.xml` file from the earlier "Describing data in an XML document" section into HTML.



```
<?xml version="1.0" encoding="utf-8"?>
<html>
  <body>
    <ul>
      <li>2 peppers</li>
      <li>1 avocado</li>
      <li>2 tomatoes</li>
      <li>1 onion</li>
    </ul>
  </body>
</html>
```

```
Stylesheet load time: 15.58 ms
Stylesheet execution time: 88.51 ms
```

Figure 4-6: XSLT templates transform XML into HTML.

Linking an XSLT file to another document

After you create your XSLT style sheet, you must associate it with your XML document. Linking the XSLT file to the XML document makes it possible for you to view the output created by the XSLT style sheet.

To associate an XSLT file with an existing XML document in Visual Studio, follow these steps:

1. **Open the XSLT style sheet in Visual Studio.**
2. **Press F4 to display the Properties window.**
3. **Click the ellipsis button for the `Input` property.**
4. **Browse to the XML document that you want to transform and then select it.**

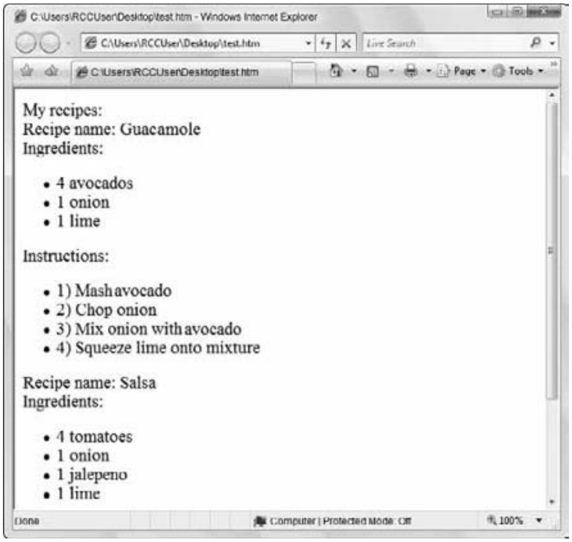
   The file's path appears in the `Input` property.
5. **Choose XML➪Show XSLT Output.**

   The output appears in a separate window.

You can place, at the top of your XML document, a directive that links the style sheet to your document. When your XML document is displayed in an XML-aware browser, the output of the style sheet appears. To add a link for the `recipe.xml` document, type the following line at the top of the `recipe.xml` file:

```
<?xml-stylesheet type="text/xsl" href="recipe.xslt"?>
```

Figure 4-7 shows the XML document displayed in Internet Explorer.


C:\Users\RCCUser\Desktop\test.htm - Windows Internet Explorer
C:\Users\RCCUser\Desktop\test.htm
Live Search
C:\Users\RCCUser\Desktop\test.htm
Page
Tools
Done
Computer | Protected Mode: Off
100%

My recipes:
Recipe name: Guacamole
Ingredients:

- 4 avocados
- 1 onion
- 1 lime

Instructions:

- 1) Mash avocado
- 2) Chop onion
- 3) Mix onion with avocado
- 4) Squeeze lime onto mixture

Recipe name: Salsa
Ingredients:

- 4 tomatoes
- 1 onion
- 1 jalepeno
- 1 lime

Figure 4-7: The XSLT output appears in the browser window.

Staying on the path with XPath

XPath is an important technology for XSLT templates. You use XPath to navigate an XML document and find the nodes, or sets of nodes, that you want to display in your output. Suppose that you have an XML document for storing recipes with a structure like this:

```
<recipes>
<recipe name="string">
    <ingredients>
      <ingredient>
        <qty>unsignedByte</qty>
        <item>string</item>
      </ingredient>
    </ingredients>
    <instructions>
      <steps ordinal="string">
        <text>string</text>
      </steps>
    </instructions>
  </recipe>
</recipes>
```

You can use an XPath expression, such as /recipes/recipe, to navigate to the recipe node in the XML document. Table 4-1 describes the XPath syntax.

Table 4-1 XPath Syntax

XPath Expression	*Syntax*	*What It Selects*
Nodename	instructions	The child nodes of the instructions node
/	/recipes	An absolute path to the root node
	/recipes/recipe	The child recipe nodes of the recipes node
//	//steps	All steps nodes, regardless of their recipe
.	/recipes/recipe/ingredients/.	The current node; the qty and item nodes in the example
..	/recipes/recipe/..	The parent node; all the nodes in the recipes node in the example
@	/recipes/recipe/@name	The name attribute of the recipe node

TIP

The easiest way to understand XPath is to use the value-of element in an XSLT style sheet and experiment with different expressions, like this:

```
<xsl:value-of select="recipes/recipe/@name" />
```

XML and .NET

The Microsoft .NET Framework provides many resources for working with XML in code. The XML-related classes in the .NET Framework have names that are similar to the XML technologies. For example, you use XPathNavigator in .NET to write code that navigates an XML document. The XML-related classes are in the System.XML namespace in .NET. Table 4-2 describes some of its common resources.

Table 4-2 System.XML Namespace in .NET

.NET Class	*What It Does*
`XMLReader`	Reads XML documents
`XMLWriter`	Writes to XML documents
`XMLDocument`	Represents an XML document
`XPathNavigator`	Navigates and edits XML documents
`XMLSchema`	Represents an XML Schema

Each of the classes in the table includes numerous properties, methods, and additional classes for working with XML documents. For more information on using XML in code, see Chapter 5 in Book V.

Chapter 5: Under the Hood with ADO.NET

In This Chapter

- **Using the .NET Framework Data Providers**
- **Connecting to data sources**
- **Managing connection strings**
- **Executing commands**
- **Accessing data with** `DataReaders`
- **Filling** `DataSets` **with** `DataAdapters`
- **Converting** `DataSets` **to XML**
- **Using ADO.NET in the real world**

Visual Studio provides many controls, designers, editors, and wizards for accessing data in Windows and Web applications. These tools are made possible by *ADO.NET,* the data access technology of the .NET Framework.

ADO.NET provides a common coding structure for accessing data, regardless of the data source. The Visual Studio data access tools, such as DataSet Designer, generate ADO.NET code behind the scenes for you. This chapter shows you how to "get under the hood" with ADO.NET and write your own ADO.NET code to access your data.

ADO is short for *ActiveX Data Objects,* the previous version of the data access technologies, before the .NET Framework was introduced.

Meet the Players

The purpose of ADO.NET is to provide a simplified model for data access, regardless of the underlying data source. By using a model like ADO.NET, developers can improve their productivity because they use one data access model — ADO.NET — to access many different kinds of data sources.

ADO.NET provides a set of common components for accessing data:

- **.NET Framework Data Provider:** Acts as a bridge between your application and a data source. Providers are available for many popular databases, including SQL Server and Oracle. When a native provider is unavailable for your data source, you use an Open Database Connectivity (ODBC) or Open Database Connectivity database (OLE DB) provider to access your data source.

  Each.NET Framework Data Provider offers a set of services for accessing data. See the following section for details on using data providers.

- `DataReader`**:** Provides forward-only access to a data source one row at a time by using one of the .NET Framework Data Providers. See the section "Reading Data with DataReaders," later in this chapter.

  `DataReaders` are a service of the .NET Framework Data Providers. `DataReader` is listed separately here because it represents an important model for data access. `DataSets` are populated with `DataReaders`.

- `DataSet`**:** Provides an in-memory cache of data that's retrieved from a data source by way of one of the .NET Framework Data Providers. See the section "Caching Data with DataSets," later in this chapter.

Figure 5-1 shows the interplay of the .NET Framework Data Providers, `DataReaders`, and `DataSets` in ADO.NET.

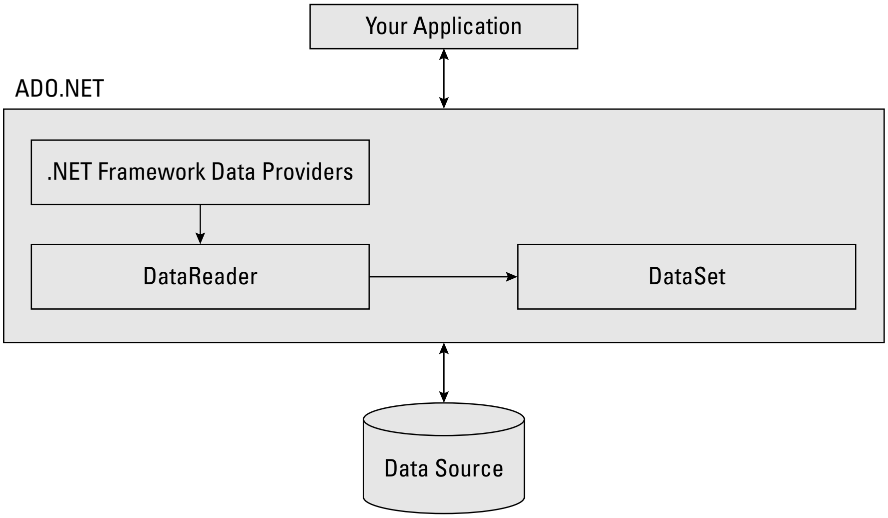


Figure 5-1: ADO.NET provides a simple model for accessing data.

You access the components of ADO.NET through the `System.Data` namespace in .NET.

The services of the .NET framework are organized into groups, or namespaces. To read more about namespaces, see Book V, Chapter 1.

Picking a Provider

ADO.NET uses data providers to provide access to many different kinds of data sources, such as SQL Server and Oracle databases. Table 5-1 summarizes the data providers available in the .NET Framework.

Table 5-1 .NET Framework Data Providers

Data Source	*.NET Namespace*	*What It Accesses*
SQL Server	`System.Data.SqlClient`	SQL Server 7.0 databases and later
OLE DB	`System.Data.OleDB`	SQL Server 6.5, Oracle, and Microsoft Access databases by using OLE DB
ODBC	`System.Data.Odbc`	SQL Server, Oracle, and Microsoft Access databases by using ODBC
Oracle	`System.Data.OracleClient`	Oracle databases by using the Oracle client connectivity software

You should always use the provider that's tuned for your data source. For example, always use the SQL Server provider to access SQL Server 7.0 databases and later and use the Oracle provider to access Oracle databases. Microsoft recommends the OLE DB provider for accessing Microsoft Access databases.

Never use a Microsoft Access database in a multitiered application. (A *multitiered application* is one in which several layers each perform different functions in the process, such as data access, business rules, and presentation.) Access isn't an enterprise-quality database and doesn't always perform well across a network. Consider using SQL Server Express Edition if you need a lightweight database.

The SQL Server provider uses its own communication channel for transmitting commands to SQL Server. As a result, the SQL Server provider is faster than the OLE DB and ODBC providers, both of which add their own layers of communication channels.

Always use the .NET Data Framework Provider for SQL Server to access SQL Server 7.0, SQL Server 2000, SQL Server 2005, and SQL Server 2008 databases.

Accessing providers

You access the .NET Framework Data Providers through their .NET namespaces. (Table 5-1 in the preceding section lists the namespaces for each of the .NET Framework Data Providers.) For example, to access the features of the SQL Server provider, type the following namespace in the Code Editor:

```
System.Data.SqlClient
```

Because of how ADO.NET is constructed, you can access data without choosing a specific data provider. For example, you can write code that retrieves data from a data source without knowing in advance whether you want to use the SQL Server provider or the Oracle provider. Not choosing a data provider is useful when you need code to be flexible enough to choose your data source at runtime.

When you access data without using a data provider, you write *provider-independent* code. When you write it, you use the `System.Data.Common` namespace rather than one of the data provider namespaces listed in Table 5-1.

Writing provider-independent data access code isn't for the faint of heart. To read more about what's involved in writing provider-independent code, look for the topic **provider independent code** in the index of the Visual Studio 2010 documentation. You can find a good resource for the topic at `http://msdn2.microsoft.com/en-us/library/t9f29wbk(VS.80).aspx`.

Objectifying the providers

Each.NET Framework Data Provider provides access to a common set of data access features and services. You use the following features and services of the providers to connect to your data source and retrieve data:

- `Connection`: Connects your application to a data source
- `Command`: Executes an SQL statement against a data source
- `DataReader`: Reads a forward-only stream of data one row at a time
- `DataAdapter`: Retrieves data for a `DataSet` and sends updates to the data source

Each .NET Framework Data Provider has its own flavor of the features and services in the preceding list. For example, the `DataAdapter` for the SQL Server data provider is the `SqlDataAdapter`.

Figure 5-2 illustrates the relationship among the data providers' features and services.

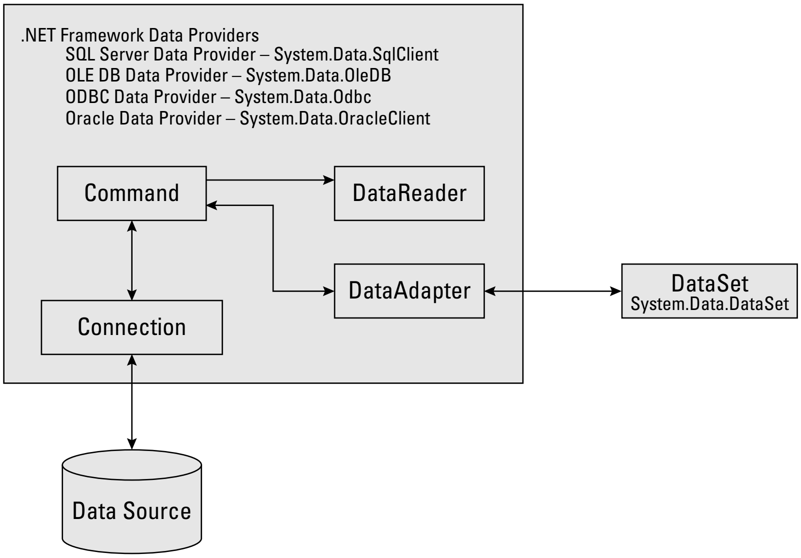

.NET Framework Data Providers
SQL Server Data Provider – System.Data.SqlClient
OLE DB Data Provider – System.Data.OleDB
ODBC Data Provider – System.Data.Odbc
Oracle Data Provider – System.Data.OracleClient
Command
DataReader
DataAdapter
Connection
DataSet
System.Data.DataSet
Data Source

Figure 5-2: Each data provider has a set of services for accessing data.

You access the features and services of the data providers through the provider's namespace. For example, you access the Oracle data provider's `DataReader`, `OracleDataReader`, at `System.Data.OracleClient`. Table 5-1 lists the namespaces for each of the data providers.

The features and services in the preceding list are *objects* of the .NET Framework Data Providers. Before you can use any of these features, you must create a new *instance* of the object you want to use. To read more about creating object instances, see Book III, Chapter 3.

Making Connections

Each of the .NET Framework Data Providers has a connection object for establishing a connection to a data source. The *connection object* is the communication pipeline from the .NET Framework Data Providers and the underlying data source to your application. Table 5-2 lists the connection objects for each data provider.

Table 5-2 .NET Framework Data Provider Connection Objects

Provider	*Connection Object*
SQL Server	`SqlConnection` `OleDbConnection`
ODBC	`OdbcConnection`
Oracle	`OracleConnection`

Connecting to a database

You use the connection object for your data provider to connect to your data source. As you can imagine, connecting to a data source is a prerequisite for retrieving data from the data source. You use a connection object to

- **Pass a connection string to the connection object.** A connection string includes your username and password for accessing the data source. The connection object sends the connection string to the data source for validation.
- **Open the connection.** Opening a connection allows you to communicate with the data source.
- **Send commands by using the connection.** Send queries to retrieve and update the data source. See the section "Using Commands," later in this chapter.
- **Close the connection.** Each connection to the data source consumes resources on the server. Always close the connection as soon as you execute your commands.

ADO.NET uses a feature called *connection pooling* that groups multiple database requests into a single connection. Connections that are exactly the same — same server, same database, same user credentials — are pooled by default with ADO.NET. Pooling connections together so that they can be reused reduces the overhead required to open and close connections.

You should leave connection pooling enabled. You can disable connection pooling if you're trying to achieve optimized performance for a specific application by using `pooling='false'` in your connection strings.

You follow the same sequence of events each time you want to access a data source. For example, to use a connection object to access an SQL Server 2008 database, follow these steps:

1. **Pick your .NET Framework Data Provider.**

   You should always use the .NET Framework Data Provider for SQL Server to access SQL Server 7.0 databases and later. You access the

SQL Server data provider by using the `System.Data.SqlClient` namespace.

2. **Create a new `SqlConnection` connection object.**

   To create the new `SqlConnection` object `MyConnection`, type the following lines of code in the Code Editor:

   - *VB*

   ```
   Dim MyConnection As New System.Data.SqlClient.SqlConnection
   ```

   - *C#*

   ```
   System.Data.SqlClient.SqlConnection MyConnection =
      new System.Data.SqlClient.SqlConnection();
   ```

3. **Create a new connection string to connect to an SQL Server 2008 database.**

   The connection string to connect to the Adventure Works sample database installed on a local instance of SQL Server 2008 Express Edition using integrated security is

   ```
   Data Source=(local)\sqlexpress;Initial Catalog=AdventureWorks;Integrated
       Security=True
   ```

   Visual Studio provides many tools for building connection strings. See the later section "Stringing up connections" for more information on creating and managing connection strings.

4. **Assign the connection string you create in Step 3 to the `Connection String` property of the `MyConnection SqlConnection` object, as shown in the following example:**

   - *VB*

   ```
   MyConnection.ConnectionString = ("Data Source=(local)\sqlexpress;Initial
       Catalog=AdventureWorks;Integrated Security=True")
   ```

   - *C#*

   ```
   MyConnection.ConnectionString = "Data Source=(local)\"+"sqlexpress;Init
       ial Catalog=AdventureWorks;"+"Integrated Security=True";
   ```

5. **Call the `Open` method in `MyConnection` to open the database connection with the following line:**

   - *VB*

   ```
   MyConnection.Open()
   ```

   - *C#*

   ```
   MyConnection.Open();
   ```

   The data provider connects to the data source by using the connection string specified in Step 4.

6. **Send commands to the database to retrieve or update data.**

   See the upcoming section "Using Commands" to see how to send commands to a data source.

7. **Call the `Close` method in `MyConnection` to close the database connection with the following line:**

   - *VB*

     ```
     MyConnection.Close()
     ```

   - *C#*

     ```
     MyConnection.Close();
     ```

Here's the entire code listing for the preceding code example:

VB:

```
Dim MyConnection As New System.Data.SqlClient.SqlConnection
MyConnection.ConnectionString = ("Data Source=(local)\sqlexpress;Initial Catalog=
    AdventureWorks;Integrated Security=True")
MyConnection.Open()
'send commands
MyConnection.Close()
```

C#:

```
  System.Data.SqlClient.SqlConnection MyConnection =
     new System.Data.SqlClient.SqlConnection();
MyConnection.ConnectionString = "Data Source=(local)\"+    "sqlexpress;Initial
    Catalog=AdventureWorks;"+ "Integrated Security=True";
MyConnection.Open();
// Send Commands
MyConnection.Close();
```

To use the connection object for any .NET Framework Data Providers, substitute the name of the provider's connection object listed in Table 5-2 where you see `SqlConnection` in the preceding code example.

For example, to work with an ODBC connection, type this line:

VB:

```
Dim MyConnection As New System.Data.Odbc.OdbcConnection
```

C#:

```
System.Data.Odbc.OdbcConnection MyConnection =
     new System.Data.Odbc.OdbcConnection();
```

Always use structured exception handling every time you open a connection. (See Book III, Chapter 7.)

Closing your connection

You should always explicitly close your connection to your data source. Closing the connection releases resources.

Opening your connections with a `Using` block is a good way to remember to always close your connections. A `Using` *block* consists of starting and ending statements that create and dispose of the resource, respectively. The following code shows a `Using` block:

VB:

```
Using resource as New resourceType
.
.
.
End Using
```

C#:

```
using( resource = new resourcetype )
{
}
```

You place, between the `Using ... End Using` statements, statements that access the resource. A database connection is an example of a resource you can use with a `Using` block, as shown in the following code sample:

VB:

```
Using MySqlConnection As New System.Data.SqlClient.
    SqlConnection()
  MySqlConnection.ConnectionString = ("Data Source=(local)\
    sqlexpress;Initial Catalog=
    AdventureWorks;Integrated Security=True")
  MySqlConnection.Open()
  'send commands
End Using
```

C#:

```
using( MySqlConnection = new System.Data.SqlClient.SqlConnection() )
{
    MySqlConnection.ConnectionString =
    "Data Source=(local)\sqlexpress;Initial Catalog="+
            "AdventureWorks;Integrated Security=True";
    MySqlConnection.Open();
    // Send Commands
}
```

There's no need to explicitly call the connection object's `Close` method with the `Using` block. The `End Using` statement automatically closes the connection for you.

Instead of creating a new, identical connection to a data source each time, ADO.NET reuses existing connections, in a process known as *connection pooling*. When you explicitly close your connection, the connection is returned to the connection pool, where it can be reused.

Stringing up connections

Nothing stops you in your tracks faster than an incorrect connection string. Without a valid connection string, your code can't establish a connection to the data source. If you're lucky enough to connect to the same data sources over and over, you have to build a working connection string only once. As long as the data source doesn't change, you can reuse your connection string.

Whether you work with the same data sources day in and day out, or are always using different data sources, Visual Studio and .NET provide many tools to help you build and manage your connection strings. See the next section for more information.

A data provider's connection object uses the connection string to establish a connection to a data source when the connection object's `Open` method is called. A *connection string* is a set of name/value pairs *(keywords)* separated by semicolons, as shown in the following example:

```
Data Source=(local)\sqlexpress;Initial Catalog=AdventureWorks;Integrated
    Security=True
```

The set of keywords used to create a connection string are determined by the data source. Common name/value pairs used to connect to the SQL Server database are described in Table 5-3.

Table 5-3 Common SQL Server Connection-String Keywords

Keyword	*What It Does*	*Usage*
`Data Source` or `Server`	Serves as name or network address of server	`server=myserver, data source=myserver\server instance, server=(local)`
`Encrypt`	Uses SSL encryption	`encrypt=true`
`Initial Catalog` or `Database`	Sets the name of the database to access	`Initial Catalog=AdventureWorks`
`Integrated Security`	Determines whether to use Windows security	`integrated security=true`
`Password` or `Pwd`	Sets the password to use when not using integrated security	`password=mypassword`
`User ID`	Sets the user ID to use when not using integrated security	`user id=myuserid`

TIP

Always use integrated security to access your data sources.

See the topic **Securing connection strings** in the Visual Studio 2010 documentation for more information. See the **Impersonation** topic in the Visual Studio 2010 documentation to read more about using integrated security with ASP.NET.

You set a connection string by using the `ConnectionString` property of the connection object.

TECHNICAL STUFF

ADO.NET uses the `ConnectionString` property to set the connection object's `DataSource` and `DataBase` properties.

Building connection strings

Visual Studio has many ways to help you build connection strings, including

- Application settings in Windows applications
- ASP.NET configuration settings
- The Add Connection dialog box
- .NET Framework Data Provider connection string builders

The .NET Framework makes extensive use of configuration setting files, such as application settings in Windows applications and configuration settings in ASP.NET, to store all kinds of information related to your application. Visual Studio provides tools for adding settings, such as connection strings, to .NET configuration files. For example, you can use the ASP.NET Web Site Administration Tool to add several predefined and custom settings to your Web site. See Book IV, Chapter 6 for more information on using the Web Site Administration Tool.

WARNING!

Don't store connection strings in your source code. Your connection strings can be extracted from compiled code, in essence enabling anyone to bypass security measures you put into place.

Adding connection strings to Windows projects

Windows projects store application settings, such as connection strings, in an XML configuration file named `app.config`.

To add connection strings to the application settings of a Windows project, follow these steps:

1. **Right-click the My Project folder in an existing Windows project to access the project's properties.**
2. **Click the Settings tab.**

   A grid appears where you enter application settings.

3. **Type** MyDbString **in the Name cell of the grid.**

4. **Select Connection String as the settings type.**

5. **Click the ellipsis button in the Value cell.**

   The Connection Properties dialog box appears.

6. **Connect to your data source by using the Connection Properties dialog box.**

   The dialog box returns a connection string to the Value cell from the Connection Properties dialog box, as shown in Figure 5-3.

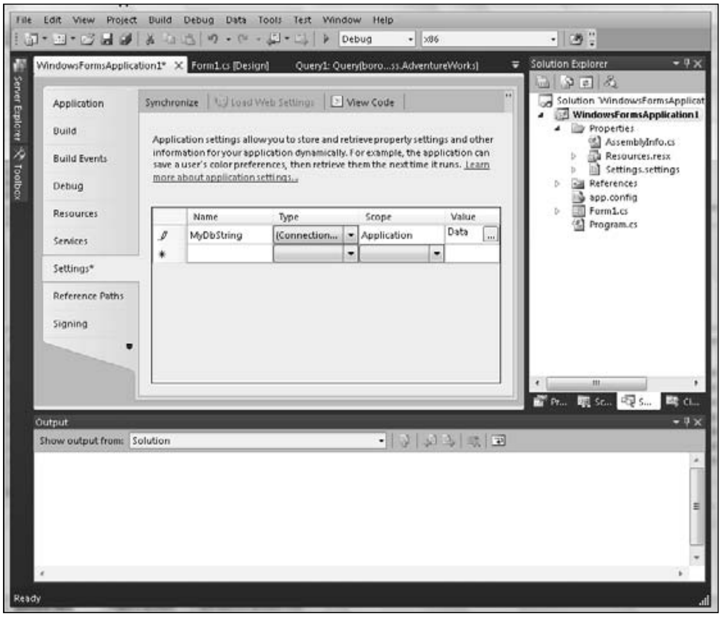


Figure 5-3: Use the Connection Properties dialog box to build your connection string.

Application settings allow you to store and retrieve property settings and other information for your application dynamically. For example, the application can save a user's color preferences, then retrieve them the next time it runs. Learn more about application settings...

	Name	Type	Scope	Value
✎	MyDbString	(Connection...	Application	Data
*				

   See the section in Book V, Chapter 2 about connecting to databases for more information on creating a connection string.

7. **Save the settings.**

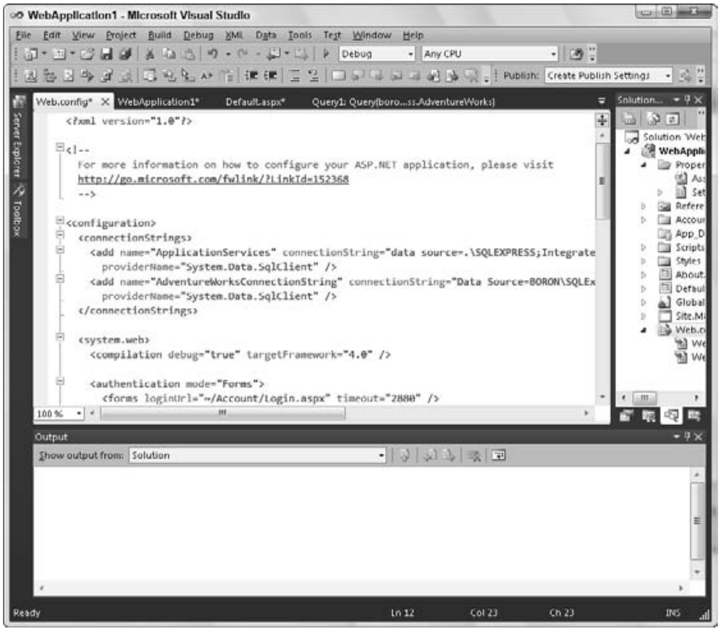
   Visual Studio creates a new `app.config` file with your connection string.

You access the connection string by using the `My.Settings` expression when writing code in VB. (Use `Properties.Settings` when writing code in C#.) See the upcoming section "Using Commands" to see an example of accessing a connection string from application settings.

Adding connection strings to Web projects

Web projects use the `web.config` file to store configuration settings, such as connection strings. The easiest way to add new connection strings to the `web.config` file is with the Data Source Configuration Wizard. (Book V, Chapter 1 walks you through using the wizard in the section about adding data to Web Forms).

You can use the Data Source Configuration Wizard to configure many different kinds of data sources. Configuring a database with the wizard prompts you to select an existing data connection or configure a new connection. After creating the configuration string, the wizard saves the connection string in the `web.config` file, as shown in Figure 5-4.



WebApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug XML Data Tools Test Window Help
Debug
Any CPU
Publish: Create Publish Settings
Web.config*
WebApplication1*
Default.aspx*
Query1: Query(boro...ss.AdventureWorks)
Server Explorer
Toolbox
Solution...
100 %
Output
Show output from: Solution
Ready
Ln 12
Col 23
Ch 23
INS

```
<?xml version="1.0"?>

<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=152368
  -->

<configuration>
  <connectionStrings>
    <add name="ApplicationServices" connectionString="data source=.\SQLEXPRESS;Integrate
      providerName="System.Data.SqlClient" />
    <add name="AdventureWorksConnectionString" connectionString="Data Source=BORON\SQLEx
      providerName="System.Data.SqlClient" />
  </connectionStrings>

  <system.web>
    <compilation debug="true" targetFramework="4.0" />

    <authentication mode="Forms">
      <forms loginUrl="~/Account/Login.aspx" timeout="2880" />
```

Figure 5-4: Use the Data Source Configuration Wizard to save connection strings in the `web.config` file.

The wizard writes the connection string to the `web.config` file. Alternatively, you can add the connection string manually to the `web.config` file. For example, a connection string for the Adventure Works database might look like this:

```
<connectionStrings>
   <add name="AdventureWorksConnectionString" connectionString="Data
     Source=(local)\sqlexpress;Initial Catalog=AdventureWorks;Integrated
     Security=True"
            providerName="System.Data.SqlClient" />
</connectionStrings>
```

Add your connection string between the `<connectionStrings></connectionStrings>` tags.

Building connection strings manually

A common theme in building connection strings is the use of the Add Connection dialog box. It's used in both the application settings for Windows and the configuration settings for Web applications.

You access the Add Connection dialog box by choosing Tools➪Connect to Database. The Add Connection dialog box creates a new connection in Server Explorer. Use Server Explorer to manage your connections to servers in your project.

You can grab the connection string from Server Explorer and reuse it elsewhere, in either code or a configuration-settings file. To copy a connection string from a data connection in Server Explorer, follow these steps:

1. **Create a new data connection by choosing Tools➪Connect to Database.**

   The Add Connection dialog box appears.

2. **Use the Add Connection dialog box to create a connection string.**

   A new data connection appears in Server Explorer, as shown in Figure 5-5.

3. **Right-click the data connection in Server Explorer.**

   You can open Server Explorer by choosing View➪Server Explorer. The keyboard shortcut for Server Explorer is Ctrl+Alt+S.

4. **Choose Properties from the shortcut menu.**

   The Properties window appears.

5. **Highlight the `ConnectionString` property.**

6. **Copy and paste the property.**

Using the ADO.NET connection string builders

Each of the .NET Framework Data Providers includes a service for building and managing connection strings. The connection string builders provide the properties needed to build a connection string. The builder outputs a properly formatted connection string that you pass to the data provider's connection object. Table 5-4 lists the connection string builders for each data provider.

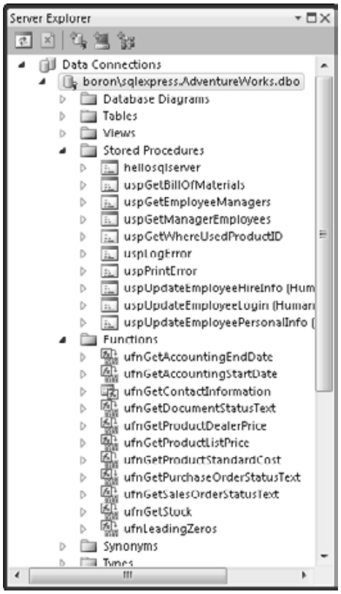


Figure 5-5: Copy and paste the `ConnectionString` property from Server Explorer.

Table 5-4 Connection String Builders

Data Provider	*Object*
SQL Server	`SqlConnectionStringBuilder`
OLE DB	`OleDbConnectionStringBuilder`
ODBC	`OdbcConnectionStringBuilder`
Oracle	`OracleConnectionStringBuilder`

The following code uses the SQL Server data provider's connection string builder, `SqlConnectionStringBuilder`, to build a connection string for the Adventure Works sample database:

VB:

```
Dim builder As New System.Data.SqlClient.SqlConnectionStringBuilder
builder.DataSource = "(local)\sqlexpress"
builder.InitialCatalog = "AdventureWorks"
builder.IntegratedSecurity = True
```

C#:

```
System.Data.SqlClient.SqlConnectionStringBuilder builder =
    new System.Data.SqlClient.SqlConnectionStringBuilder();
builder.DataSource = "(local)\sqlexpress";
builder.InitialCatalog = "AdventureWorks";
  builder.IntegratedSecurity = True;
```

You pass the connection string from `SqlConnectionStringBuilder` to the connection object's `ConnectionString` property like this:

VB:

```
Dim MyConnection As New System.Data.SqlClient.SqlConnection
MyConnection.ConnectionString = builder.ConnectionString
```

C#:

```
System.Data.SqlClient.SqlConnection MyConnection =
     new System.Data.SqlClient.SqlConnection();
MyConnection.ConnectionString = builder.ConnectionString;
```

Using Commands

You use a .NET Framework Data Provider's `Command` object to execute queries after connecting to a data source. Each of the data providers has a `Command` object. To use a `Command` object, follow these steps:

1. **Associate the `Command` object with a connection object.**

   Use the `Connection` property to make the association, like this:

   - *VB*

     ```
     MySqlCommand.Connection = MySqlConnection
     ```

   - *C#*

     ```
     MySqlCommand.Connection = MySqlConnection;
     ```

   The `Connection` object is the communication pipeline between the command and the data source.

2. **Specify an SQL statement. Use the `Command` object `CommandText` property to set the SQL statement.**

   For example, to set the SQL statement for an ODBC data provider `Command` object, type

   - *VB*

     ```
     MyOdbcCommand.CommandText = "SELECT * FROM CUSTOMER"
     ```

   - *C#*

     ```
     MyOdbcCommand.CommandText = "SELECT * FROM CUSTOMER";
     ```

3. **Call one of the `Command` object's `Execute` methods.**

   The `Execute` method runs the SQL statement specified in the `Command` object `CommandText` property. The `Command` object has three `Execute` methods:

   - `ExecuteReader`: Returns a `DataReader` object. Use `ExecuteReader` any time you want fast access to a forward-only stream of data.
   - `ExecuteScalar`: Returns a single value. Use `ExecuteScalar` when you know that your query will return only one value, such as a `SELECT Count(*)` query.
   - `ExecuteNonQuery`: Doesn't return any rows. Use `ExecuteNonQuery` any time you need to execute a query, such as a `CreateTable` statement, that doesn't return any rows.

   You must call the `Open` method of the data provider's connection object before you call the `Command` object `Execute` method.

Most data-centric applications make extensive use of stored procedures for data access. To use a stored procedure with a `Command` object, you must

- Set the `Command` object `CommandType` property to `StoredProcedure`.
- Use the `Command` object `Parameters` collection to define the stored procedure's input and output parameters.

To use the following stored procedure to access an SQL Server database by using the SQL Server data provider, follow these steps:

1. **Create the `SqlConnection` object and set the object's `ConnectionString` property, as shown in the following example:**

   - *VB*

   ```
   Dim MySqlConnection As New System.Data.SqlClient.SqlConnection
   MySqlConnection.ConnectionString = My.Settings.MyDbString
   ```

   - *C#*

   ```
   System.Data.SqlClient.SqlConnection MySqlConnection =
        new System.Data.SqlClient.SqlConnection();
   MySqlConnection.ConnectionString =
        Properties.Settings.MyDbString;
   ```

   The `ConnectionString` property accesses the `MyDbString` connection string from the project's `app.config` file. See the earlier section "Adding connection strings to Windows projects" for more information about the `app.config` file.

2. **Create the `SqlCommand` object by entering the following lines:**
   - *VB*

     ```
     Dim MySqlCommand As New System.Data.SqlClient.SqlCommand
     ```
   - *C#*

     ```
     System.Data.SqlClient.SqlCommand MySqlCommand =
          new System.Data.SqlClient.SqlCommand();
     ```
3. **To set the `SqlCommand` object `Connection` property, enter this line:**
   - *VB*

     ```
     MySqlCommand.Connection = MySqlConnection
     ```
   - *C#*

     ```
     MySqlCommand.Connection = MySqlConnection;
     ```
4. **Set the `CommandText` property of the `SqlCommand` object to the name of the stored procedure you want to execute, as the following lines show:**
   - *VB*

     ```
     MySqlCommand.CommandText = "uspGetEmployeeManagers"
     ```
   - *C#*

     ```
     MySqlCommand.CommandText = "uspGetEmployeeManagers";
     ```
5. **Set the `CommandType` property to `StoredProcedure` by entering the following line:**
   - *VB*

     ```
     MySqlCommand.CommandType = CommandType.StoredProcedure
     ```
   - *C#*

     ```
     MySqlCommand.CommandType = CommandType.StoredProcedure;
     ```
6. **Create a new `SqlParameter` object, like this:**
   - *VB*

     ```
     Dim MyParameter As New System.Data.SqlClient.SqlParameter
     ```
   - *C#*

     ```
     System.Data.SqlClient.SqlParameter My Parameter =
          new System.Data.SqlClient.SqlParameter();
     ```
7. **To set the `ParameterName` and `Value` properties for the `SqlParameter`, assign values to the properties as shown:**
   - *VB*

     ```
     MyParameter.ParameterName = "@EmployeeID"
     MyParameter.Value = "6"
     ```

- *C#*

  ```
  MyParameter.ParameterName = "@EmployeeID";
  MyParameter.Value = "6";
  ```

8. **Add the `SqlParameter` object to the `SqlCommand Parameters` collection, like this:**

   - *VB*

     ```
     MySqlCommand.Parameters.Add(MyParameter)
     ```

   - *C#*

     ```
     MySqlCommand.Parameters.Add(MyParameter);
     ```

9. **Call the `Open` method of the `SqlConnection` object:**

   - *VB*

     ```
     MySqlConnection.Open()
     ```

   - *C#*

     ```
     MySqlConnection.Open();
     ```

10. **Call the `Command` object's `Execute` method with the following statement:**

    - *VB*

      ```
      MySqlDataReader = MySqlCommand.ExecuteReader()
      ```

    - *C#*

      ```
      MySqlDataReader = MySqlCommand.ExecuteReader();
      ```

    The `Execute` method passes the stored procedure from the `Command` object to the database by using the connection object.

The `Command` object has a `CommandBuilder` you can use to automatically generate `Command` objects for single-table data access. See the topic **CommandBuilder object** in the Visual Studio 2010 documentation.

Reading Data with DataReaders

ADO.NET `DataReaders` are old-school data-access services. You use a `DataReader` any time you need fast, forward-only access to your data. Unlike a `DataSet`, which retrieves your data into an in-memory database model, there's no storage mechanism with a `DataReader`. When you retrieve data by using a `DataReader`, you had better have your "catcher's mitt" open to store the data.

TECHNICAL STUFF

A `DataReader` is often called a *firehose cursor*.

Each of the .NET Framework Data Providers provides a DataReader. Table 5-5 lists the DataReader objects for each provider.

Table 5-5 DataReader Objects

Provider	*DataReader Object*
SQL Server	SqlDataReader
OLE DB	OleDbDataReader
ODBC	OdbcDataReader
Oracle	OracleDataReader

You use the ExecuteReader method of the Command object to retrieve data for a DataReader.

Using a DataReader to retrieve data from a data source involves these steps:

1. **Call the Command object's ExecuteReader method.**

   ExecuteReader builds the DataReader.

2. **Call the DataReader Read method to advance to the next record.**

   The first time you call Read, the next record is the first record because the DataReader is positioned in front of the first record.

3. **Use the DataReader Get accessors to retrieve data from the row of data.**

   The DataReader retrieves one row at a time.

   See the section "Retrieving data with the Get accessors," later in this chapter, to see the DataReader Get accessors in action.

4. **Advance to the next record by using the Read method.**

   Because the DataReader retrieves one row at a time from the data source, use common practice to execute the Read method by using a loop.

   See the following section to see the DataReader used with a while loop.

Even though you need only two methods to use the DataReader — ExecuteReader and Read — a lot of setup work is involved. The following steps walk you through using a DataReader to retrieve records:

1. **Select a .NET Framework Data Provider, as described in the section "Picking a Provider," earlier in this chapter.**

2. **Create a new connection object for the data provider, as described in the earlier section "Making Connections."**

3. **Declare a new variable of the `DataReader` object type for your data provider.**

   For example, to declare a new `DataReader` variable for the ODBC data provider, you would type the following:

   - *VB*

     ```
     Dim odbcReader As System.Data.Odbc.OdbcDataReader
     ```

   - *C#*

     ```
     System.Data.Odbc.OdbcDataReader odbcReader =
          new System.Data.Odbc.OdbcDataReader();
     ```

   Notice that you aren't using the `New` keyword to create a new instance of the `DataReader` object. There are no constructors for `DataReaders`. Calling the `Command` object `ExecuteReader` method builds the `DataReader`.

4. **Create a new `Command` object for your data provider to retrieve data for the object, as described in the preceding section.**

5. **Call the `Open` method of the connection object to establish a connection to the data source.**

6. **Call the `ExecuteReader` method of the `Command` object you create in Step 4 and pass the results to the `DataReader` object you create in Step 2.**

   For example, to call the `ExecuteReader` method on an `OdbcCommand` object and pass the results to an `OdbcDataReader` object you create in Step 3, type the following:

   - *VB*

     ```
     odbcReader = odbcCommand.ExecuteReader()
     ```

   - *C#*

     ```
     odbcReader = odbcCommand.ExecuteReader();
     ```

   The `OdbcCommand` object executes the query against the `OdbcConnection` object when `ExecuteReader` is called and builds the `OdbcDataReader` object.

7. **Call the `Read` method of the `DataReader` object to retrieve one record from the data source.**

   For example, to call the Read method for the `OdbcDataReader` you create in Step 3, type the following:

   - *VB*

     ```
     odbcReader.Read()
     ```

- *C#*

  ```
  odbcReader.Read();
  ```

  The `OdbcDataReader` object advances to the next record.

  The default position of the `DataReader` is before the first row in the result set. The `Reader` method advances the `DataReader` to the next record.

8. **Retrieve values from the `DataReader`.**

   See the section “Retrieving data with the Get accessors,” later in this chapter, for more information about using the `DataReader Get` accessors to retrieve data.

9. **Close your connection object, as described in the section “Closing your connection,” earlier in this chapter.**

   TIP

   Close the `Connection` object by calling the `Close` method of the connection object or using a `Using` block.

Here’s the entire code listing:

VB:

```
Dim odbcConnection As New System.Data.Odbc.OdbcConnection
Dim odbcReader As System.Data.Odbc.OdbcDataReader
Dim odbcCommand As New System.Data.Odbc.OdbcCommand
odbcConnection.ConnectionString = My.Settings.MyOdbcConnectionString
odbcCommand.Connection = odbcConnection
odbcCommand.CommandText = "SELECT * FROM CUSTOMER"
odbcConnection.Open()
odbcReader = odbcCommand.ExecuteReader()
odbcReader.Read()
'do something here with the data in the row
odbcConnection.Close()
```

C#:

```
System.Data.Odbc.OdbcConnection odbcConnection =
     new System.Data.Odbc.OdbcConnection();
System.Data.Odbc.OdbcDataReader odbcReader =
     new System.Data.Odbc.OdbcDataReader();
System.Data.Odbc.OdbcCommand idbcCommand =
     new System.Data.Odbc.OdbcCommand();
odbcConnection.ConnectionString =
     Properites.Settings.MyOdbcConnectionString;
odbcCommand.Connection = odbcConnection;
odbcCommand.CommandText = "SELECT * FROM CUSTOMER";
odbcConnection.Open();
odbcReader = odbcCommand.ExecuteReader();
odbcReader.Read();
//do something here with the data in the row
odbcConnection.Close();
```

Stepping through data

The DataReader Read method advances the DataReader to the next row in the result set. As long as more rows are present, the Read method returns the value True. The Read method is typically used to test the Read method's return value in a while loop.

A while loop loops through a set of statements as long as a test condition remains True. You can use a while loop to iterate through each row in a DataReader's result set and perform the same action on each row. For example, to use a while loop to step through a SqlDataReader object named MySqlDataReader, type the following lines:

VB:

```
While (MySqlDataReader.Read() = True)
   Me.lstDepartments.Items.Add(MySqlDataReader("Name"))
End While
```

C#:

```
while( MySqlDataReader.Read() )
{
     lstDepartments.Items.Add(
       Convert.ToString(MySqlDataReader["Name"]));
}
```

The preceding example gets the value in column Name of the MySqlDataReader result set and adds the value to the items collection of a lstDepartments list box. The while loop performs the statement inside the loop as long as the Read method returns True. You end up with a list box full of items from the SqlDataReader. See the next section, about how to get data from a row.

TIP

Use the HasRows property of the DataReader to test whether the DataReader has more rows. The HasRows property returns the value True if more rows are present.

Retrieving data with the Get accessors

The DataReader object provides access to a forward-only result set which presents one row at a time. When you're deciding how to work with the data in a row, you have to consider whether you want to work with the data

- In its native format or by using a typed accessor
- In a single column in the row or all the columns in the row

`DataReader` provides several `Get` accessors for retrieving data. You should retrieve the data by using a typed accessor, such as `GetDateTime`.

The data types used in the underlying data source aren't the same as the data types used in the .NET Framework. The typed accessors convert the value from its native database type to a .NET Framework type.

`DataReader` provides several options for retrieving data by using typed accessors. Each `DataReader` provides a set of common typed `Get` accessors, such as

- `GetChar`: Retrieves data as a `char` data type
- `GetDateTime`: Retrieves data as a `DateTime` data type
- `GetInt16`: Retrieves data as an `Int16` data type
- `GetString`: Retrieves data as a `string` data type

`SqlDataReader` provides special types that work exclusively with SQL Server database types. Examples include

- `GetSqlChars`
- `GetSqlDateTime`
- `GetSqlInt16`
- `GetSqlString`

Use `GetSql` typed accessors when you're using the `SqlDataReader`. `GetSql` typed accessors are more precise than the .NET data types.

If you're unsure of the column's native data type, query the data source. `DataReaders` provide the `GetFieldType` and `GetDataTypeName` methods you can use.

Of course, you can also retrieve data in its native format. `DataReader` provides several methods, such as

- `GetValue` and `GetValues`
- `Item`

When you access the data in its native format, you must ensure that the data is converted to the appropriate .NET Framework data type. For example, the following statement retrieves a column by using the `Item` method and uses the `ToString` method to convert it to a string:

VB:

```
myString = MySqlDataReader.Item("Name").ToString()
```

C#:

```
myString = MySqlDataReader.Item["Name"].ToString();
```

To read more about converting data types, see Book III, Chapter 2. Additionally, see the topic **mapping data types** in the Visual Studio 2010 help documentation to see how native data types are mapped to .NET Framework data types.

Most `Get` accessors use a zero-based column index to retrieve a column. A zero-based index starts counting elements at zero instead of at one. For example, in a table with the columns CustID, FirstName, and LastName, the FirstName column might have the column index of one. To access the FirstName column by using the `GetSqlString` accessor, you type the following line:

VB:

```
MySqlDataReader.GetSqlString(1)
```

C#:

```
MySqlDataReader.GetSqlString(1);
```

Using column indexes to retrieve values may be fast for the `DataReader`, but it's slow and confusing to a programmer. Fortunately, `DataReaders` have two methods for accessing columns by name:

- `Item`: Returns the column's value in its native format
- `GetOrdinal`: Returns the index number of the column

Each of these approaches has its drawbacks. If you use the `Item` method, you must explicitly convert the column to a .NET data type. Using `GetOrdinal` hits the data source twice — once to get the ordinal and again to retrieve the data by using the ordinal.

Here's an example of using `GetOrdinal` to access data:

VB:

```
Dim nameCol, groupNameCol As Integer
nameCol = MySqlDataReader.GetOrdinal("Name")
```

```
groupNameCol = MySqlDataReader.GetOrdinal("GroupName")
While (MySqlDataReader.Read() = True)
  myString = String.Format("{0} {1}", MySqlDataReader.
    GetString(nameCol), MySqlDataReader.
    GetString(groupNameCol))
  Me.lstDepartments.Items.Add(myString)
End While
```

C#:

```
int nameCol, groupNameCol;
nameCol = MySqlDataReader.GetOrdinal("Name");
groupNameCol = MySqlDataReader.GetOrdinal("GroupName");
while( MySqlDataReader.Read())
{
  myString = String.Format("{0} {1}", MySqlDataReader.
    GetString(nameCol), MySqlDataReader.
    GetString(groupNameCol));
  lstDepartments.Items.Add(myString)
}
```

TIP

Use the GetValues method to retrieve all the columns in a row at one time. The GetValues method requires you to pass in an array that the method fills with the columns from the row.

Here's an example of using the GetValues method:

VB:

```
Dim MyArray(MySqlDataReader.FieldCount - 1) As Object
MySqlDataReader.GetValues(MyArray)
```

C#:

```
Object[] MyArray = new Object[MySqlDataReader.FieldCount-1];
MySqlDataReader.GetValues(MyArray);
```

In the first line of this example, you use the DataReader FieldCount property to set the size of the array. In the second line, you pass the array to the GetValues method. The array is filled with the values for the entire row.

The GetValues method retrieves values in their native data formats. The values must be converted to .NET data types.

With the values in the array, you can access them by using the properties and methods of arrays. To read more about using arrays, see Book III, Chapter 2.

The GetValues method, which is a fast way to grab an entire row, is typically used by GetValues to pass the array to the ItemArray method of the DataRow object.

DataRows are rows in a DataTable. You specify ahead of time which columns and data types exist in the DataTable. When you add a new row by using ItemArray, the data is plugged into the columns and converted to the column's data type. The following code fills an array and passes the array to a new DataRow in the myTable DataTable:

VB:

```
MySqlDataReader.GetValues(array)
row = myTable.NewRow()
row.ItemArray = array
```

C#:

```
MySqlDataReader.GetValues(array);
row = myTable.NewRow();
  row.ItemArray = array;
```

The values in the array are converted to the appropriate data type for each column in the DataRow.

Retrieving schema info

The DataReader includes a GetSchemaTable method that you can use to retrieve the schema information about the result set. The GetSchemaTable method returns a DataTable, as shown in the following code:

VB:

```
Dim table As New DataTable
table = MySqlDataReader.GetSchemaTable()
```

C#:

```
DataTable table = new DataTable():
  table = MySqlDataReader.GetSchemaTable();
```

You can use the DataRow's ItemArray method, as described in the preceding section, to add new rows to a table built with GetSchemaTable.

Caching Data with DataSets

DataSets are an important element in data access. ADO.NET provides the DataSet as a built-in memory cache for storing data retrieved from a data source. See the section about understanding DataSets in Book V, Chapter 3.

The workhorse behind the DataSet is the DataAdapter. It provides the following services to the DataSet:

- Populates the DataSet with data from a data source by using the Fill method
- Updates a data source with changes made in the DataSet by using the Update method

Each of the .NET Framework Data Providers has a DataAdapter object. Table 5-6 lists the DataAdapter object for each data provider.

Table 5-6 DataAdapter Objects

Data Provider	*DataAdapter Object*
SQL Server	SqlDataAdapter
Ole DB	OleDbDataAdapter
ODBC	OdbcDataAdapter
Oracle	OracleDataAdapter

The DataAdapter object uses a set of Command objects to send SQL statements to a data source. The DataAdapter object exposes the Command objects by using the following set of properties:

- SelectCommand

  You must specify a SelectCommand before you can call the DataAdapter Fill method.
- InsertCommand
- UpdateCommand
- DeleteCommand

See the earlier section “Using Commands” to read more about the properties of Command objects.

Filling a DataSet

A DataAdapter acts as a bridge between the DataSet and the data source. Filling a DataSet involves these tasks:

- Create a DataAdapter and a DataSet.
- Create a Command object that holds the SELECT statement or stored procedure to retrieve data from the data source.
- Set the DataAdapter SelectCommand property to the Command object.
- Call the DataAdapter Fill method and pass in the DataSet.

As is the case with all the data access features of ADO.NET, filling a DataSet isn't as simple as creating a few objects and then calling the Fill method. You have to complete a number of prerequisite steps, such as creating a connection object.

The following example walks you through using a DataAdapter to fill a DataSet:

1. **Pick the .NET Framework Data Provider best suited for your data source, as described in the earlier section "Picking a Provider."**

   This example uses the SQL Server data provider to fill a DataSet by using data from the Adventure Works sample database in SQL Server 2008.

2. **Create a new connection and connection string, as described earlier in this chapter, in the section "Making Connections."**

   Enter the following lines of code:

   - *VB*

     ```
     Dim MySqlConnection As New System.Data.SqlClient.SqlConnection
     MySqlConnection.ConnectionString = My.Settings.MyDbString
     ```

   - *C#*

     ```
     System.Data.SqlClient.SqlConnection MySqlConnection =
          new System.Data.SqlClient.SqlConnection();
         MySqlConnection.ConnectionString =
        Properties.Settings.MyDbString;
     ```

3. **Create a new `DataAdapter` and `DataSet`, as shown in the following example:**

   - *VB*

     ```
     Dim MySqlDataAdapter As New System.Data.SqlClient.SqlDataAdapter()
     Dim MyDataSet As New System.Data.DataSet
     ```

- *C#*

```
System.Data.SqlClient.SqlDataAdapter MySqlDataAdapter =
     new System.Data.SqlClient.SqlDataAdapter();
System.Data.DataSet MyDataSet =
     new System.Data.DataSet();
```

4. **Create a new `Command` object, associate the `Command` object with your connection object, and set the `CommandText` property, as described in the earlier section "Using Commands."**

   For example, the following code sample creates a new `SqlCommand` object, associates the object with an `SqlConnection` object, and sets the `SqlCommand` object to an SQL statement:

   - *VB*

```
Dim MySelectCommand As New System.Data.SqlClient.SqlCommand
MySelectCommand.Connection = MySqlConnection
MySelectCommand.CommandText = "SELECT * FROM HumanResources.Department"
```

   - *C#*

```
System.Data.SqlClient.SqlCommand MySelectCommand =
     new System.Data.SqlClient.SqlCommand();
MySelectCommand.Connection = MySqlConnection;
    MySelectCommand.CommandText =
       "SELECT * FROM HumanResources.Department";
```

5. **Set the `DataAdapter`'s `SelectCommand` property to the `Command` object you create in Step 4, as shown in the following:**

   - *VB*

```
MySqlDataAdapter.SelectCommand = MySelectCommand
```

   - *C#*

```
MySqlDataAdapter.SelectCommand = MySelectCommand;
```

6. **Call the `DataAdapter Fill` method and pass in the `DataSet` as a parameter:**

   - *VB*

```
MySqlDataAdapter.Fill(MyDataSet)
```

   - *C#*

```
             MySqlDataAdapter.Fill(MyDataSet);
```

Behind the scenes, the `Fill` method does the following:

- ✦ Retrieves data from the data source by passing the SQL statement specified in the `SelectCommand` to a `DataReader`.
- ✦ Creates a `DataTable` by using the column information from the data source and adds the rows to the `DataTable`.
- ✦ Adds the `DataTable` to the specified `DataSet`.

Figure 5-6 shows the relationship among the objects of the SQL Server data provider used in the preceding example. The relationships are similar for other .NET Framework data providers.

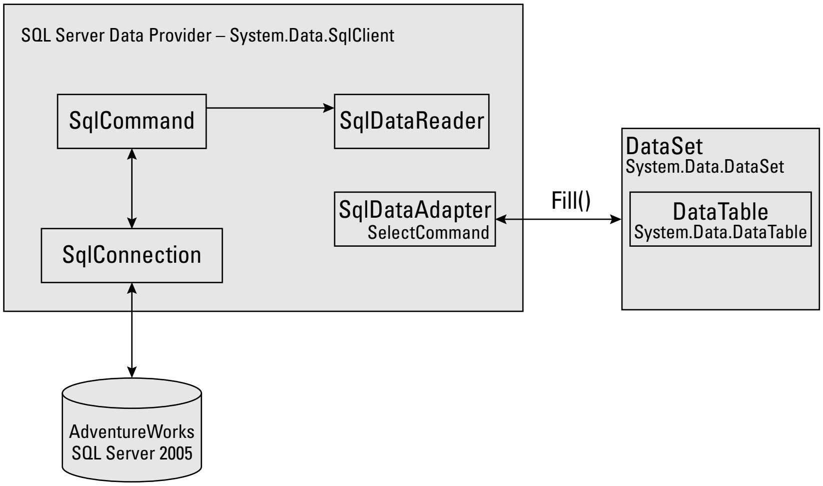


Figure 5-6: The SQL Data Adapter is the bridge between the data source and the `DataSet`.

Updating data with the Update method

You call the `DataAdapter` `Update` method to send data updates from the `DataSet` to the data source. The `Update` method uses `InsertCommand`, `UpdateCommand`, and `DeleteCommand` to update the data source.

The `Update` method requires you to pass in a `DataSet`, a `DataTable`, or an array of `DataRows`, as shown in the following example:

VB:

```
MySqlDataAdapter.Update(MyDataSet)
```

C#:

```
MySqlDataAdapter.Update(MyDataSet);
```

The `DataAdapter` uses the `Command` objects specified in its properties to execute SQL statements against the data source.

Each of the .NET Framework Data Providers has a `CommandBuilder` object that you can use to automatically generate commands against a single-table data source. `CommandBuilder` automatically builds the commands for a `DataAdapter` to use.

To use `SqlCommandBuilder` with the example from the preceding section, follow these steps:

1. **Create a new `SqlCommandBuilder` object after Step 5 in the preceding section.**

   - *VB*

   ```
   Dim builder As New System.Data.SqlClient.SqlCommandBuilder()
   ```

   - *C#*

   ```
   System.Data.SqlClient.SqlCommandBuilder builder =
        new System.Data.SqlClient.SqlCommandBuilder();
   ```

2. **Set the `SqlCommandBuilder` `DataAdapter` property:**

   ```
   builder.DataAdapter = MySqlDataAdapter
   ```

   The `SqlCommandBuilder` uses the `DataAdapter` `SelectCommand` property to build the `Insert`, `Update`, and `Delete` commands.

3. **Call the `Update` method instead of the `Fill` method in Step 6 in the preceding section.**

Using TableAdapters

`TableAdapter`, a new data access feature in ADO.NET, encapsulates all the retrieve and update commands and a connection object for a single table. Visual Studio provides extensive support for creating `TableAdapters` by using the TableAdapter Configuration Wizard in DataSet Designer.

You can call the commands of a generated `TableAdapter` by using the standard “dot” notation of IntelliSense in the Code Editor. To access the properties and methods of a generated `TableAdapter`, follow these steps:

1. **Use DataSet Designer to create a new `DataSet` with the Department table from the Adventure Works database.**

   See the section about exploring DataSet Designer in Book V, Chapter 3 for more information on creating `DataSets`.

2. **Open a Windows Form and then drag and drop the Department table from the Data Sources window.**

   A `DepartmentTableAdapter` is created.

3. **Double-click the form to access the form’s `Load` event.**

   The Code Editor appears.

4. **Type the following code in the Code Editor:**

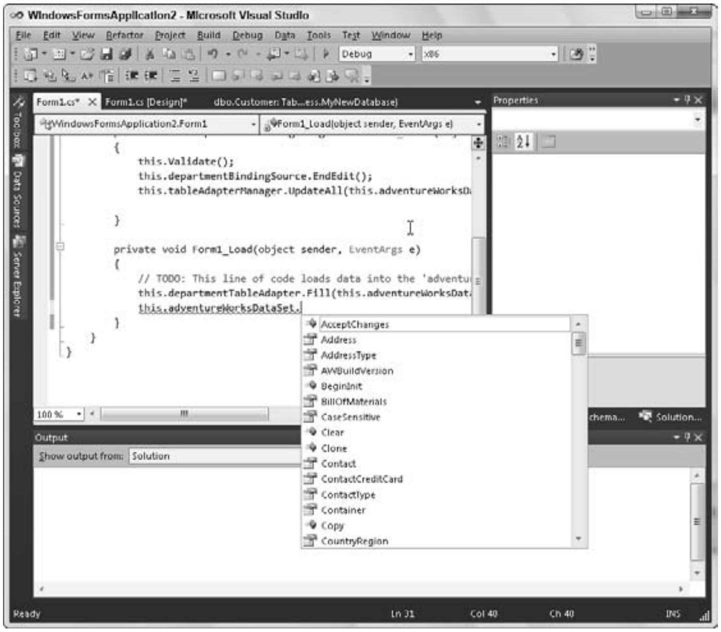
   - *VB*

   ```
   Me.DepartmentTableAdapter.
   ```

- *C#*

  `DepartmentTableAdapter.`

  The Code Editor displays a list of properties and methods available for `DepartmentTableAdapter`, as shown in Figure 5-7.



WindowsFormsApplication2 - Microsoft Visual Studio
File Edit View Refactor Project Build Debug Data Tools Test Window Help
Debug
x86
Form1.cs* Form1.cs [Design]* dbo.Customer: Tab...ess.MyNewDatabase)
WindowsFormsApplication2.Form1
Form1_Load(object sender, EventArgs e)
Properties
Toolbox
Data Sources
Server Explorer
100 %
AcceptChanges
Address
AddressType
AWBuildVersion
BeginInit
BillOfMaterials
CaseSensitive
Clear
Clone
Contact
ContactCreditCard
ContactType
Container
Copy
CountryRegion
chema...
Solution...
Output
Show output from: Solution
Ready
Ln 31
Col 40
Ch 40
INS

```
        {
            this.Validate();
            this.departmentBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.adventureWorksD

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the 'adventu
            this.departmentTableAdapter.Fill(this.adventureWorksDat
            this.adventureWorksDataSet.
        }
    }
}
```

Figure 5-7: Use IntelliSense to access the methods and properties of `TableAdapters` generated by the designers in Visual Studio.

5. **Select a property or method from the list.**

Use the TableAdapter Configuration Wizard to generate `TableAdapters`.

Using transactions

SQL statements are executed by databases in transactions. A *transaction* is a group of statements executed against a database. You use statements such as `BEGIN TRANSACTION` and `COMMIT WORK` to mark the start and end of a transaction. Between the start and end are SQL statements that retrieve and modify data. The `ROLLBACK WORK` statement rolls back, or undoes, the statements executed in the transaction.

The .NET Framework provides a new model for transaction processing with the `System.Transactions` namespace. You use it to create two kinds of database transactions:

✦ **Implicit transactions:** Use the `TransactionScope` object to encapsulate a block of ADO.NET code in a transaction, as shown in this code:

- *VB*

```
Using scope As New System.Transactions.TransactionScope()
.
.
.
End Using
```

- *C#*

```
using(scope = new System.Transactions.TransactionScope())
{
.
.
.
}
```

✦ **Explicit transaction:** Create a `CommittableTransaction` object where you specifically call the object's `Commit` and `Rollback` methods. The following code creates a new `CommittableTransaction` object:

- *VB*

```
Dim tx As New System.Transactions.CommittableTransaction
```

- *C#*

```
          System.Transactions.CommittableTransaction txt =
                new System.Transactions.CommittableTransaction();
```

You must add a reference to `System.Transactions` before you can access the objects in the namespace.

The `System.Transactions` namespace is the model for all kinds of transactions in .NET, not just database transactions. (See the Visual Studio 2010 documentation.)

Supporting XML with ADO.NET

ADO.NET provides extensive support for XML. You can use ADO.NET to do the following:

✦ **Fill a `DataSet` by using an XML document.** Call the `DataSet` `ReadXml` method to populate a `DataSet` with an XML document.

✦ **Create or infer a `DataSet` schema from an XML Schema definition.** Call a `DataSet` `ReadXmlSchema` or `InferXmlSchema` methods to create the `DataSet` schema.

✦ **Create an XML document or XML Schema from a `DataSet`.** Call the `DataSet` `GetXml` method to write the `DataSet` content as an XML document. Call `GetXmlSchema` to write an XML Schema file from the `DataSet` schema.

- **Synchronize a `DataSet` contents with an XML document.** Use the `XmlDataDocument` object to create an XML document with the data from a `DataSet`.

The following sample code creates a `DataSet` and populates it with data from an XML document with the name `recipe.xml`:

VB:

```
Dim ds As New System.Data.DataSet
ds.ReadXml("recipe.xml", XmlReadMode.InferSchema)
```

C#:

```
System.Data.DataSet ds = new System.Data.DataSet();
  ds.ReadXml("recipe.xml", XmlReadMode.InferSchema);
```

Using the `DataSet` from the preceding code example, write an XML Schema to the `recipe.xsd` file with this code:

VB:

```
ds.WriteXmlSchema("recipe.xsd")
```

C#:

```
  ds.WriteXmlSchema("recipe.xsd");
```

You can combine the extensive ADO.NET support for XML with SQL Server 2008 support for XML. Using SQL Server 2008, you can

- Use the `xml` data type to store entire XML documents or fragments.
- Associate XML Schemas with `xml` data types to create typed XML.
- Retrieve data stored in relational tables as XML markup using the `FOR XML` clause.
- Retrieve XML data as relational data by using the `OPENXML` function.

The Adventure Works sample database provides several examples of using the `xml` data type. To read more about using XML with SQL Server 2008, see the topic **Using XML in SQL Server** in the SQL Server 2008 Books Online documentation.

Using ADO.NET in Your Applications

Using the features described in this chapter, you can use ADO.NET to access data seven ways from Sunday. Couple these features with all the designers in

Visual Studio, and you have even more choices. Here are some recommendations for accessing data with ADO.NET:

- ✦ **Populate custom data entities with `DataReaders`.** Many developers create their own custom data types for storing the data entities their application uses. For example, you can create a `Customer` data type and a `CustomerAddress` data type. Use a `DataReader` to populate your `Customer` and `CustomerAddress` data types. You can use this approach to separate the application from its underlying data source. (See Book III, Chapter 2.)
- ✦ **Use typed `DataSets` for prototyping or in conjunction with other data storage.** Although typed `DataSets` are very fast to build, they perform slower than custom data entities in your application. They can be especially slow in Web applications. You don't have to use a typed `DataSet` for all your data access — you can use a combination of approaches. Typed `DataSets` are a good way to quickly build prototype applications.
- ✦ **Create data access class libraries.** Whatever approach you use, usually you should encapsulate your data access methods in a separate class file or class library. You create public methods or properties that return `DataSets`, `DataTables`, hashtables, arrays, or custom data entities that your data controls consume. See Book III, Chapter 2 for more information on hashtables and arrays.

  For example, here's a method signature that returns a `DataTable` from the class `DataAccess`:

  - *VB*

    ```
    Public Function GetDepartments() As System.Data.DataTable
    ```

  - *C#*

    ```
    public System.Data.DataTable GetDepartments()
    {
    }
    ```

  The `GetDepartments` method encapsulates all the data access code populating the `DataTable`.

  To use the `DataTable` as a data source for a `BindingSource` component, enter these lines:

  - *VB*

    ```
    Dim data As New DataAccess
    Me.MyBindingSource.DataSource = data.GetDepartments
    ```

  - *C#*

    ```
    DataAccess data = new DataAccess();
    MyBindingSource.DataSource = data.GetDepartments();
    ```

Use the BindingSource component as the data source for a databound control.

- **Evaluate whether to use stored procedures or ad hoc SQL queries.** People are on both sides of the camp on this issue. Ad hoc queries don't automatically make the sky fall.
- **Extend generated typed `DataSets`.** Use partial classes to add features to typed `DataSets`. For example, you may decide to use a `DataReader` to populate a simple lookup table.
- **Create your own helper classes.** A great deal of repetition occurs in building data access code. You can create your own helper classes, though, to cut down on all the repetition.
- **Use the Enterprise Library.** Another way to deal with all the repetition involved in coding data access is to use the Enterprise Library. It has all the best practices of using ADO.NET baked right in.

Chapter 6: Using Visual Studio with SQL Server

In This Chapter

- **Installing SQL Server 2008 Express Edition**
- **Configuring the Adventure Works sample database**
- **Creating scripts in database projects**
- **Creating stored procedures in SQL Server projects**
- **Using Visual Database Tools**

Most useful programs require some kind of data storage. Often, that's a database program like Microsoft SQL Server. SQL Server 2008 is the latest release of the popular database program and ships with Visual Studio 2010.

This chapter shows you how to install SQL Server 2008 and how to use Visual Studio to create and manage database objects.

Getting Acquainted with SQL Server 2008

SQL Server 2008 is the latest version of the popular Microsoft database management system. Each edition of Visual Studio includes a license for one of the editions of SQL Server 2008. Table 6-1 lists the editions of SQL Server and describes how to get the program.

Table 6-1 SQL Server Editions

SQL Server Version	*When to Use It*	*How to Get It*
Enterprise Edition	In large companies	Purchase separately; retail price $24,999
Standard Edition	In small- and medium-size businesses	Purchase separately; retail price $5,999
Workgroup Edition	In departments or branch offices	Purchase separately; retail price $3,899

(continued)

Table 6-1 *(continued)*

SQL Server Version	*When to Use It*	*How to Get It*
Express Edition	For prototypes or local development	Free download from Microsoft Web site; included with all Visual Studio Express Editions and Visual Studio Standard Editions
Developer Edition	For prototypes or local development	Included with Professional, Tools for Office, and Team Editions of Visual Studio
Compact Edition	In a database for mobile devices	Free download from Microsoft Web site
Web Edition	For Web hosters and end customers deploying publicly-facing web Applications	$15 per process per month

TIP

Get more information about each of the SQL Server 2008 editions, including pricing and product comparisons, on Microsoft's SQL Server portal at

`http://www.microsoft.com/sqlserver/2008/en/us/pricing.aspx`

You can download SQL Server Express Edition and Mobile Edition from the portal at h`ttp://www.microsoft.com/downloads/details.aspx?FamilyID=220549b5-0b07-4448-8848-dcc397514b41`.

Most developers use one of these two editions of SQL Server:

- **Express Edition** is free to use and redistribute. Express Edition replaces Microsoft SQL Server 2000 Desktop Engine (MSDE), the lightweight SQL Server 2000 database. You can include a local copy of a database in your application and freely distribute Express Edition with your application. You can easily use Express Edition as the database for a small Web site, for example. The size of Express Edition databases is limited to 4GB. Express Edition is also a good choice for developers who don't want or need the overhead of Developer Edition.
- **Developer Edition** is a fully functioning version of SQL Server 2008 Enterprise Edition. A Developer Edition license restricts you to using the product only in development and testing environments. Developer Edition is a good choice for consultants and corporate developers who may need to access complex databases.

TIP

Unless you know that you need the features of Developer Edition, start out with Express Edition and then upgrade if you need it.

The examples in this book use SQL Server Express Edition; many developers always install Developer Edition. SQL Server 2008 Express Edition is installed by default with Visual Studio 2010. However, you'll likely want to install SQL Server 2008 Express Edition with Advanced Tools so that you can use the SQL Server Management Console to configure your databases.

Installing SQL Server Express Edition

SQL Server 2008 Express Edition with Advanced Tools is installed separately from Visual Studio 2010.

TIP

Install Visual Studio 2010 first and then install SQL Server 2008 Express Edition. Express Edition requires version 3.5 of the .NET Framework, which is installed as part of the SQL Server 2008 installation.

Preparing your machine to install SQL Server requires many steps. Your installation steps depend on the services and components you choose to install.

Follow these steps to install SQL Server Express Edition:

1. **Install Visual Studio 2010 or version 3.5 of the .NET Framework.**

   See Book II, Chapter 1 for more information about installing Visual Studio 2010 Professional.

2. **Launch the SQL Server Express Edition setup by running the installer setup file.**

   The Start page appears.

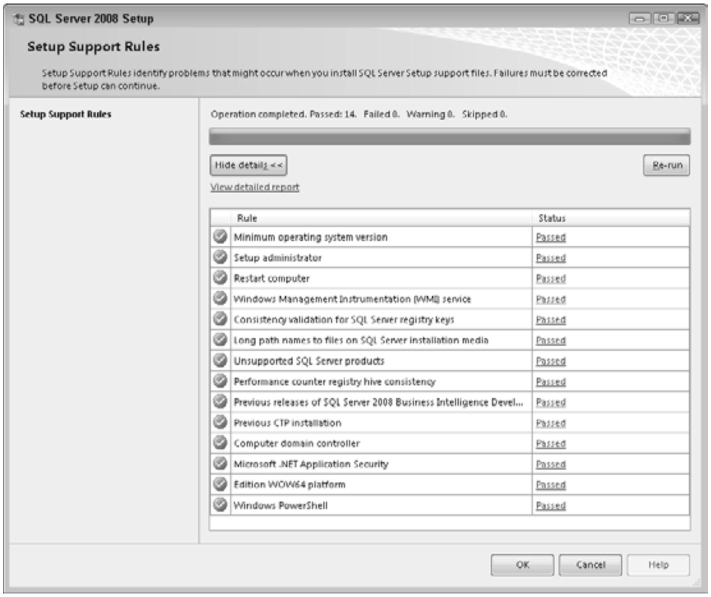
3. **Before installing SQL Server, review the hardware and software requirements and read the release notes.**

   The hardware and software requirements include instructions for accessing the Books Online documentation, where you can find complete installation instructions.

4. **After reviewing the installation documentation, click the link to install the server.**

   The Setup Support Rules page appears as shown in Figure 6-1. Correct any errors that appear and click OK.

   The wizard checks for hardware and software requirements.



Setup Support Rules

Setup Support Rules identify problems that might occur when you install SQL Server Setup support files. Failures must be corrected before Setup can continue.

Setup Support Rules

Operation completed. Passed: 14. Failed 0. Warning 0. Skipped 0.

	Rule	Status
✓	Minimum operating system version	Passed
✓	Setup administrator	Passed
✓	Restart computer	Passed
✓	Windows Management Instrumentation (WMI) service	Passed
✓	Consistency validation for SQL Server registry keys	Passed
✓	Long path names to files on SQL Server installation media	Passed
✓	Unsupported SQL Server products	Passed
✓	Performance counter registry hive consistency	Passed
✓	Previous releases of SQL Server 2008 Business Intelligence Devel...	Passed
✓	Previous CTP installation	Passed
✓	Computer domain controller	Passed
✓	Microsoft .NET Application Security	Passed
✓	Edition WOW64 platform	Passed
✓	Windows PowerShell	Passed

Figure 6-1: Review the results of the system configuration check.

5. **Enter the product key and click Next.**

   The wizard displays the License Terms.

6. **Accept the License Terms and click Next.**

7. **Click Install.**

   The wizard installs the prerequisites.

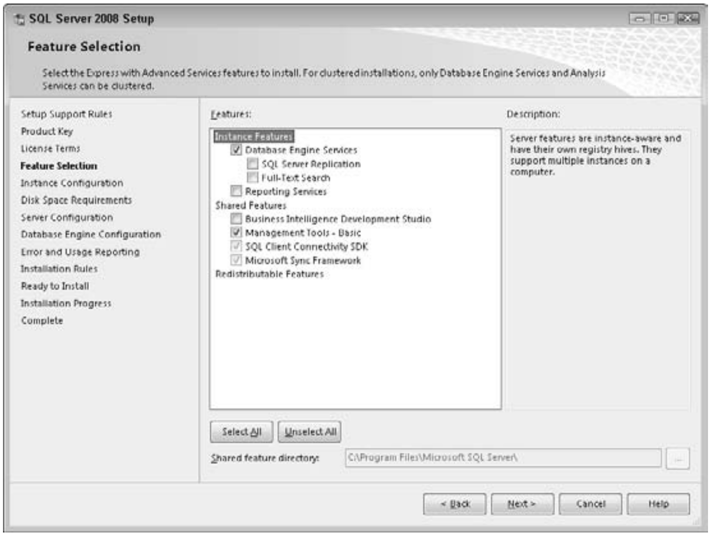
8. **Click Next.**

   The wizard displays the Feature Selection page.

9. **Select the features you want to install, as shown in Figure 6-2.**

   The components you can choose to install include

   - *Database Engine Services*: Installs the database engine that allows you to create and use databases
   - *Reporting Services*: Installs tools to support reporting
   - *Shared Features*: Installs the client tools and documentation you need to administer SQL Server 2008 and develop database applications



Feature Selection

Select the Express with Advanced Services features to install. For clustered installations, only Database Engine Services and Analysis Services can be clustered.

Features:

Instance Features
- ☑ Database Engine Services
  - □ SQL Server Replication
  - □ Full-Text Search
- □ Reporting Services

Shared Features
- □ Business Intelligence Development Studio
- ☑ Management Tools - Basic
- ☑ SQL Client Connectivity SDK
- ☑ Microsoft Sync Framework

Redistributable Features

Description:

Server features are instance-aware and have their own registry hives. They support multiple instances on a computer.

Shared feature directory: C:\Program Files\Microsoft SQL Server\

Figure 6-2: The SQL Server Feature Selection page.

At a minimum, you should choose to install Database Engine Services and Shared Features. The remaining services allow you to create specialized database applications. You should install them as you need them.

Unless you need the more advanced features of Developer Edition, you may find that Express Edition works fine. You can download SQL Server 2008 Express Edition with Advanced Services at

```
http://msdn2.microsoft.com/en-us/express/bb410792.aspx
```

10. **Click Next.**

    The Disk Space Requirements page appears.

11. **Click Next.**

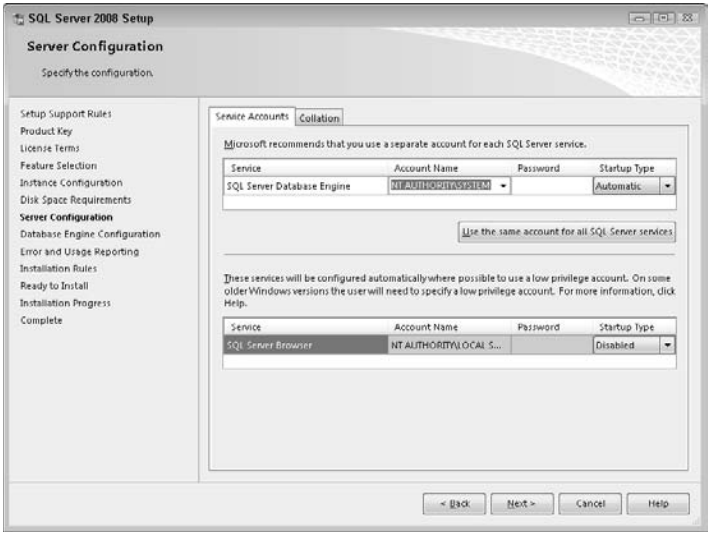
    The Server configuration page appears.

12. **Select the account name to use for SQL Server, as shown in Figure 6-3, and click Next.**

    The Database Engine Configuration page appears.

13. **Click Next.**

    The Error and Usage Reporting page appears.



Microsoft recommends that you use a separate account for each SQL Server service.

Service	Account Name	Password	Startup Type
SQL Server Database Engine	NT AUTHORITY\SYSTEM		Automatic

These services will be configured automatically where possible to use a low privilege account. On some older Windows versions the user will need to specify a low privilege account. For more information, click Help.

Service	Account Name	Password	Startup Type
SQL Server Browser	NT AUTHORITY\LOCAL S...		Disabled

Figure 6-3: The SQL Server Account Configuration page.

14. **Click Next.**

    The Installation Rules page appears.

15. **Click Next.**

    The Ready to Install page appears. Here, you can review the features of SQL server you're installing.

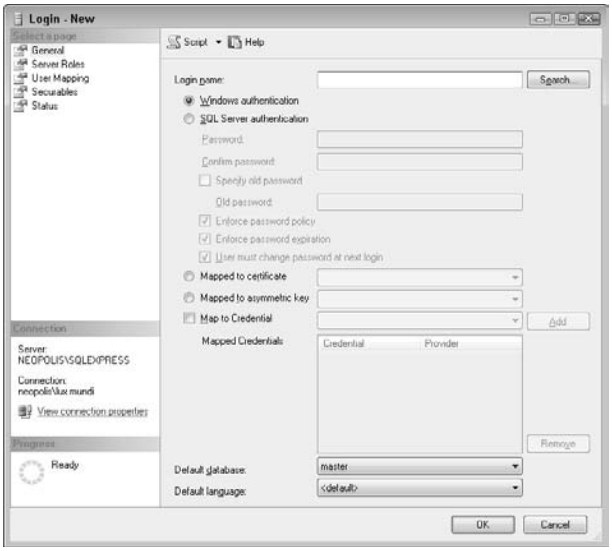
16. **Click Install to begin the installation.**

TIP

Use the Add or Remove Programs utility in Windows Control Panel to add components to, or remove them from, your SQL Server installation.

TIP

To add users that will have adequate rights and permissions to administer SQL server, you can run the Microsoft SQL Server Management Studio that was installed with SQL Server 2008. To add a new select the Security node in the tree view on the left side of the window, select the Logins node. Right-click this node and choose New Login from the contextual menu that appears. The New Login dialog box, shown in Figure 6-4, appears.



Login name:

● Windows authentication
○ SQL Server authentication

Password:

Confirm password:

□ Specify old password

Old password:

☑ Enforce password policy
☑ Enforce password expiration
☑ User must change password at next login

○ Mapped to certificate
○ Mapped to asymmetric key
□ Map to Credential

Server: NEOPOLIS\SQLEXPRESS

Connection: neopolis\lux mundi

Ready

Default database: master

Default language: <default>

Figure 6-4: Run the SQL Server Management Studio to add users that can administer SQL Server.

Working with the SQL Server tools

SQL Server 2008 includes many helpful tools for working with databases. These tools include

- **SQL Server Management Studio:** Create and manage database objects.
- **Books Online:** Browse the comprehensive SQL Server documentation.
- **Sample databases and code:** Use samples to figure out how things are supposed to work.

If you decide to use SQL Server 2008 Express Edition, you need to install some additional tools to make your life easier. You should install the following tools:

- Adventure Works sample database
- Books Online
- Management Studio Express

You can download the Adventure Works sample database at the Microsoft Codeplex download page for SQL Server 2008 Product Samples at

```
http://msftdbprodsamples.codeplex.com/releases/view/24854
```

You can download Books Online for SQL Server 2008 from Microsoft at

```
http://www.microsoft.com/downloads/details.aspx?FamilyId=765433F7-0983-4D7A-B628-
    0A98145BCB97&displaylang=en
```

TIP

After installation, you can use SQL Server 2008's policy-based management tools found on the Management node in Server Management Studio. These tools help you disable unused services and features of SQL Server so that you can minimize your security exposure.

You're likely to use the SQL Server Management Studio frequently for administering your databases. You can launch the Management Studio by choosing Start➪All Programs➪Microsoft SQL Server 2008. Figure 6-5 shows a Management Studio pane.

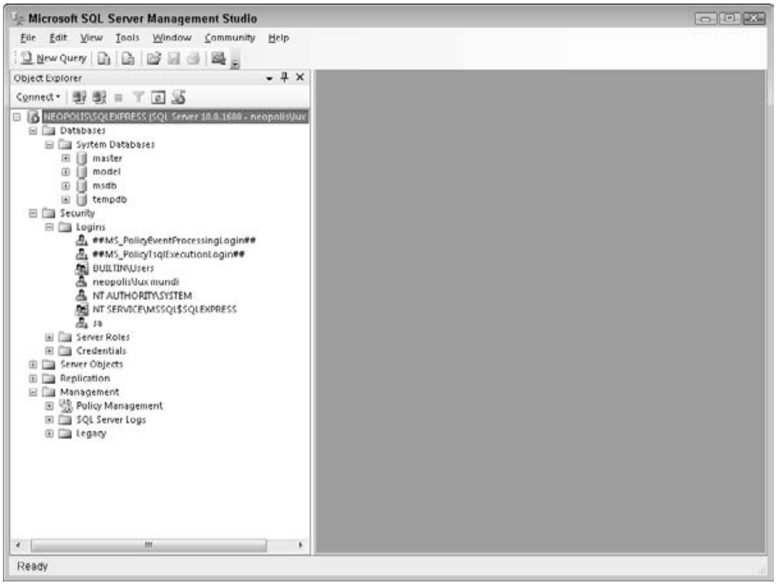


Figure 6-5: Use the SQL Server Management Studio to administer your databases.

The Management Studio features Object Explorer, which is similar to Server Explorer in Visual Studio. Like in Server Explorer, you right-click an object in Object Explorer to display a list of commands you can execute, as shown in Figure 6-6.

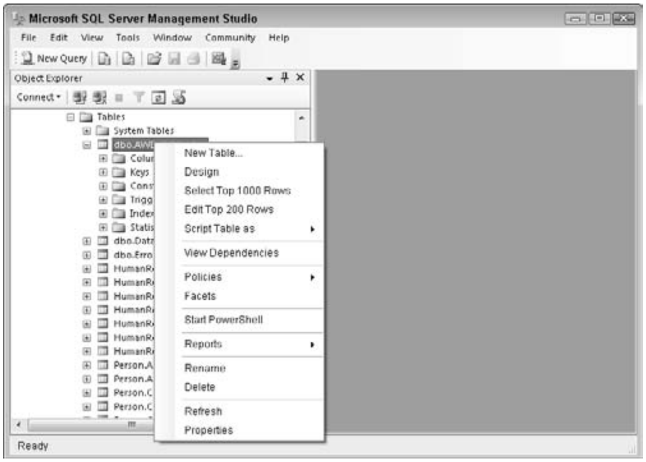


Figure 6-6: Right-click an object to display a list of commands.

The Management Studio uses many of the same designers and explorers as Visual Studio. For example, Query and View Designer is the same. See the section "Managing Your Database with Visual Database Tools," later in this chapter, to see Query and View Designer in action.

One cool feature of the Management Studio is the `Script Table As` command: Right-click a table in Object Explorer and choose Script Table As from the contextual menu. Select the kind of script you want to create, such as `CREATE`, and Management Studio generates the script.

Use the Management Studio or Visual Studio for creating queries. SQL Server 2008 doesn't include a separate query tool, such as the SQL Server 2000 Query Analyzer.

The Management Studio for SQL Server Developer Edition includes support for solutions and projects. You use Solution Explorer similarly to how you use Visual Studio to work with database projects that store script items and database connections.

Using the Adventure Works sample

The Adventure Works sample database showcases many of the new features of SQL Server 2008. One feature that Adventure Works uses extensively is *schemas,* which allow database objects to be grouped together. For example, the Adventure Works database uses a schema named HumanResources. The tables Employee and Department are associated with the HumanResources schema.

One drawback of using schemas is that more typing is required when you use SQL statements to access data. For example, to access the Employee table, you must type **HumanResources.Employee**.

The code samples you install with SQL Server 2008 provide a set of scripts you can use to either remove the schemas or create synonyms for the schemas. The scripts are

- AlterSchemaToDbo.sql
- CreateSynonymsDbo.sql

Both of these scripts have the same net effect: You can access the tables without specifying a schema first. The main difference is that if you use the first script, AlterSchemaToDbo.sql, the code samples included with SQL Server don't work. If you don't care about using code samples, use whichever script you want.

You can reverse either script by using the scripts AlterSchemaFromDbo.sql or DropSynonymsDbo.sql.

The scripts are available at

```
C:\Program Files\Microsoft SQL Server\100\Samples\Engine\Administration\
    AdventureWorks\Scripts
```

Work around this problem by executing one of the scripts in this section. You can work around the problem by manually editing the SQL statements; see the section about adding data to Web Forms in Chapter 1 of Book V.

Creating Database Projects

You can create two kinds of database projects in Visual Studio, and they are vastly different:

- **Database Project:** Create and manage scripts by using SQL.
- **SQL Server Project:** Create database objects and retrieve and update data from databases by using .NET programming languages such as Visual Basic and C#.

In an SQL Server Project, you use Visual Basic or C# to write complex data access procedures that take advantage of the features of the .NET Framework. Database projects are containers for storing SQL scripts.

Managing scripts with database projects

Visual Studio provides a project template for database projects. You use database projects to

- Create SQL scripts.
- Store database references.
- Place scripts under source control.
- Run and test scripts.

Storing scripts in a project keeps them together in a single container and allows you to place those scripts under source control. You can use the Visual Studio Visual Database Tools to generate scripts.

For more information about these tools, see the section "Managing Your Database with Visual Database Tools," later in this chapter. To read more about source control, see Book VI, Chapter 3.

You can add any kind of SQL script to a database project. Visual Studio provides templates and tools for generating certain kinds of scripts, such as

- Change and Create scripts
- Stored procedures
- Triggers
- Tables
- Views
- Database queries

To create a database project, follow these steps:

1. **Press Ctrl+Shift+N to open the New Project dialog box.**
2. **Click the plus sign (+) to expand the Other Project Types line.**
3. **Click the Database project type.**

   A list of project templates appears.
4. **Click the SQL Server 2008 Database Project icon.**
5. **Type a name and location for the project.**

6. **Click OK.**

   Visual Studio prompts you to select a database reference.

   The Add Database Reference dialog box appears if Visual Studio finds existing database connections in Server Explorer. The New Database Reference dialog box appears if no existing database connections are in Server Explorer.

7. **Select an existing database reference or create a new one.**

   Visual Studio creates the database project and adds the database reference to Solution Explorer

   Adding a new database reference is similar to creating a new database connection. See the section about connecting to databases in Chapter 2 of Book V for more information.

   See the next section for more information about working with database references.

Referencing databases

Database projects use database references to know which database to execute an SQL script against. Database references appear in Solution Explorer and are saved with the database project.

An SQL Server project uses a single database reference for the entire project. Other projects, such as Windows and Web projects, use database *connections* rather than database *references.* See the section about connecting to databases in Chapter 2 in Book V.

A database reference uses a database connection to know how to connect to the database. Database connections are visible in Server Explorer. You use Server Explorer to browse a database connection and view database objects, such as tables and views. You can't browse database objects by using a database reference in Solution Explorer. However, you can use Server Explorer to browse the database connection to which a database reference points.

When you add a new database reference to a database project, Visual Studio checks to see whether any existing database connections are in Server Explorer. If so, it prompts you to select those connections.

Otherwise, Visual Studio prompts you to add a new database reference. Adding a new one creates a new data connection in Server Explorer. The New Database Reference dialog box functions exactly the same as the Add Connection dialog box, which you use to add new data connections in Server Explorer.

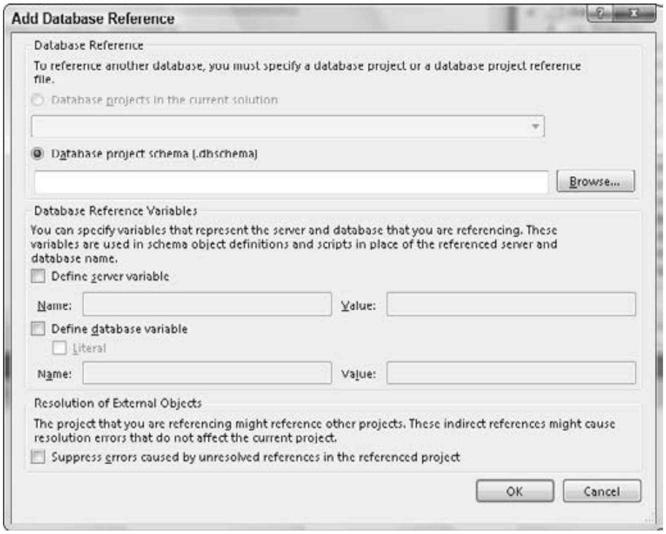
See the section about connecting to databases in Chapter 2 of Book V for more information on creating database connections in Server Explorer.

A database project can have multiple database references. To add a database reference to an existing database project, follow these steps:

1. **Right-click Database References in Solution Explorer.**
2. **Choose New Database Reference from the contextual menu.**

   The Add Database Reference dialog box appears, or the New Database Reference dialog box appears if no database connections exist in Server Explorer for the project.
3. **Select an existing reference or create a new one.**

   The Add Database Reference dialog box displays a list of all existing data connections in Server Explorer, as shown in Figure 6-7. If there are no existing data connections, the Add References dialog box doesn't appear.



Add Database Reference

Database Reference

To reference another database, you must specify a database project or a database project reference file.

○ Database projects in the current solution

● Database project schema (.dbschema)

Database Reference Variables

You can specify variables that represent the server and database that you are referencing. These variables are used in schema object definitions and scripts in place of the referenced server and database name.

□ Define server variable

Name: Value:

□ Define database variable

□ Literal

Name: Value:

Resolution of External Objects

The project that you are referencing might reference other projects. These indirect references might cause resolution errors that do not affect the current project.

□ Suppress errors caused by unresolved references in the referenced project

Figure 6-7: The Add Database Reference dialog box displays a list of existing data connections from Server Explorer.

4. **Click the Add New Reference button if you don't see the data connection you need.**
5. **Click OK.**

   Visual Studio adds the database reference to the Solution Explorer.

Database references allow you to store references to multiple databases. For example, use a reference to a local database when you're working with a test database. Use another reference to a production database when you're ready to execute your scripts in a live environment.

Set your test database as the default database reference. All your scripts are executed against the default database reference. To run your scripts against another database, you must explicitly select the database each time you execute the script. To set a database as the default database, right-click the database reference in Solution Explorer and then choose Set As Project Default from the contextual menu.

Creating scripts

Database projects include five project templates for creating SQL scripts. The available templates are

- Stored Procedure Script
- SQL Script
- Table Script
- Trigger Script
- View Script

The templates provide boilerplate SQL statements for creating scripts. For example, the Table Script template has a `CREATE TABLE` statement. The SQL Script template is a blank template.

Create your own SQL Script templates and save them as project item templates.

To create a new script in a database project, follow these steps:

1. **Right-click the database project folder in Solution Explorer.**
2. **Choose Add SQL Script from the contextual menu.**

   The Add New Item dialog box appears.

   Alternatively, choose Add Existing Item from the contextual menu to add an existing SQL script.
3. **Click the icon for the kind of SQL script you want to create.**
4. **Type a name for the script.**
5. **Click the Add button.**

   Visual Studio adds the script to the database project. The script appears in the Script Editor.

You can use the Visual Studio Visual Database Tools to generate scripts and queries. Scripts that you can generate include

- Change scripts
- Create scripts
- Database queries

See the upcoming section "Managing Your Database with Visual Database Tools" to read more about using the Visual Database Tools to generate scripts.

Scripts created with Visual Studio use the file extension `.sql`. Because there are so many naming conventions for SQL scripts, Visual Studio recognizes more than just the `.sql` extension. An abbreviated list of file extensions appears in Table 6-2.

Table 6-2 File Extensions

File Extension	*File Type*
`.tab`	Table definition
`.prc`	Stored procedure
`.viw`	View
`.trg`	Trigger
`.udt`	User-defined data type

Using the script editor

Visual Studio includes a script editor for creating SQL scripts. The editor doesn't feature IntelliSense, like other Visual Studio editors such as the Code Editor, do.

Executing scripts

You can execute an entire script or portions of a script, depending on what you want to accomplish. SQL scripts are often quite lengthy. If you don't need to run the whole script, you can simply highlight the portion of the syntax you need to run. Of course, you can always let the whole script run from start to finish.

To execute a script from a contextual menu, follow these steps:

1. **Right-click the script in the script editor or Solution Explorer.**
2. **Choose Execute SQL from the contextual menu.**

   The SQL script executes against the default database. The output appears in the Database Results window.

Follow these steps to run a portion of an SQL script that's open in the script editor:

1. **Highlight the SQL statement you want to execute in the script editor.**
2. **Right-click the selected text.**
3. **Choose Run Selection from the contextual menu.**

To execute a script against a specific database reference, follow these steps:

1. **Right-click the script file in Solution Explorer.**
2. **Choose Execute SQL from the contextual menu.**

   The Run On window appears.

TIP

Drag and drop a script file onto the database reference that you want to execute against in Solution Explorer.

Handling data with SQL Server projects

Previous versions of Visual Studio featured integration between the Visual Studio toolset and SQL Server. Integration was limited to using wizards and visual designers for manipulating database objects. Although that level of integration still exists, the Visual Studio SQL Server projects allow you to access SQL Server 2008 in a way that wasn't possible until now.

SQL Server projects are a new project type in Visual Studio designed to take advantage of integration between SQL Server 2008 and the .NET Framework. With SQL Server projects, you can do any of the following tasks:

- Write and debug .NET code that performs complex data operations by using the Code Editor.
- Deploy .NET assemblies as database objects to SQL Server 2008 databases.
- Execute .NET assemblies in SQL Server 2008.

REMEMBER

Code written using the languages of .NET, such as C# and Visual Basic, is compiled into .NET assemblies. Code that takes advantage of the services of the Common Language Runtime (CLR) is *managed* code.

Using SQL Server projects, you write stored procedures, views, triggers, and other database objects by using .NET code. You execute your code using SQL statements to call the database objects.

Managed code that executes in SQL Server 2008 is a *CLR routine.* CLR routines are sometimes also described using the word *managed,* such as managed stored procedure or managed trigger.

To create and execute a managed stored procedure named `GetEmployees`, follow these steps:

1. **Use .NET code to create a data access method named `GetEmployees`.**

   Assume that the method uses the namespace

   ```
   MyDataProcedures.GetEmployeesProc.GetEmployees
   ```

   It doesn't matter what the code inside `GetEmployees` actually does. Presumably, the code accesses data, but it could just as easily add 2 + 2.

2. **Compile the code into the .NET assembly `GetEmployees.dll`.**

   Because .NET assemblies use the file extension `.dll`, they're often referred to as DLL files.

3. **Deploy the `GetEmployees.dll` to an SQL Server 2008 database.**

   You can choose Build➪Deploy Solution in Visual Studio to deploy assemblies to SQL Server 2008 databases in SQL Server projects.

   Alternatively, connect to your SQL Server 2008 database by using the Management Studio and execute the following SQL statement in a new query:

   ```
   CREATE ASSEMBLY GetEmployees from 'GetEmployee.dll' WITH PERMISSION_SET
       = SAFE
   ```

4. **Create a stored procedure with the following SQL statement to execute the .NET assembly:**

   ```
   CREATE PROCEDURE uspGetEmployees
   AS
   EXTERNAL NAME MyDataProcedures.GetEmployeesProc.GetEmployees
   ```

   Visual Studio performs this step for you when you use the `Deploy Solution` command.

5. **Type the following SQL statement in a new query in SQL Server Management Studio to execute the stored procedure:**

   ```
   EXEC uspGetEmployees;
   ```

   SQL Server 2008 executes the code in the .NET assembly `GetEmployees`.

You use an SQL Server project to create .NET assemblies that you can deploy to SQL Server 2008. To create a new SQL Server project, follow these steps:

1. **Press Ctrl+Shift+N to open the New Project dialog box.**
2. **Click the plus sign (+) next to the Visual Basic project type.**
3. **Click the Database project type.**

   A list of database project templates appears.
4. **Click the Visual Basic SQL CLR Database Project icon.**
5. **Type a name and location for the project.**
6. **Click OK.**

   The Add Database Reference dialog box appears, or the New Database Reference dialog box appears if no data connections exist.
7. **Select an existing database reference or create a new reference.**

   Visual Studio creates the new SQL Server project.

Unlike a database project, an SQL Server project can reference only a single database. See the earlier section "Managing scripts with database projects" for more information on creating database references.

To change the database that an SQL Server project references, follow these steps:

1. **Double-click My Project in Solution Explorer.**

   The project's settings appear in the document window.
2. **Click the Database tab.**

   The database reference appears.
3. **Click the Browse button.**

   The Add Database Reference dialog box appears.
4. **Select an existing database reference or create a new reference.**

Enabling integration with SQL Server 2008

At the heart of SQL Server projects lies the integration of SQL Server 2008 with the CLR of the .NET Framework. SQL CLR integration allows you to deploy and execute .NET code to your SQL Server.

Before you can execute .NET code on your SQL Server, you must enable integration between SQL Server and the Common Language Runtime. To enable SQL CLR integration, execute the following SQL statement using the SQL Server Management Studio:

```
sp_configure 'clr enabled', 1
GO
RECONFIGURE
GO
```

Create a script file in a database project to execute the preceding statement or use the SQL Server Management Studio.

Saying, "Hello, SQL Server"

SQL Server projects provide templates for creating common database objects, such as

- Stored procedures
- Triggers
- User-defined functions
- User-defined types
- Aggregates

You add new items to the project by using the Add New Item dialog box. To create a stored procedure that outputs `Hello SQL Server!`, follow these steps:

You must set your project to target the .NET 3.5 CLR version when building a database project for SQL Server 2008.

1. **Right-click the project in Solution Explorer.**
2. **Choose Add➪Stored Procedure from the contextual menu.**

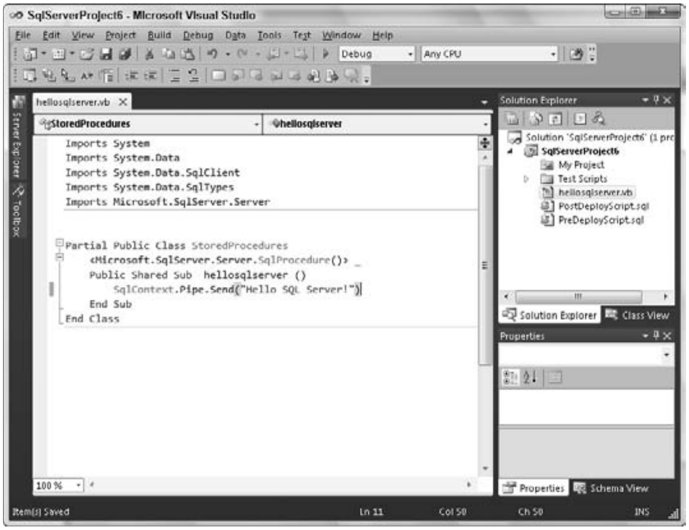
   The Add New Item dialog box appears.
3. **Type the name** hellosqlserver.vb**.**
4. **Click the Add button.**

   The stored procedure appears in the Code Editor.
5. **Replace the line `Add your code here` with the following line:**

   ```
   SqlContext.Pipe.Send("Hello SQL Server!")
   ```

   Figure 6-8 shows you the code. Notice that you don't have to write any SQL statements to create the stored procedure. Visual Studio creates the stored procedure to access your .NET code when you deploy the project.
6. **Choose Build➪Build Solution.**

   Visual Studio compiles the project.



```
Imports System
Imports System.Data
Imports System.Data.SqlClient
Imports System.Data.SqlTypes
Imports Microsoft.SqlServer.Server


Partial Public Class StoredProcedures
    <Microsoft.SqlServer.Server.SqlProcedure()> _
    Public Shared Sub  hellosqlserver ()
        SqlContext.Pipe.Send("Hello SQL Server!")
    End Sub
End Class
```

Figure 6-8: Create a stored procedure in the Visual Studio Code Editor.

7. **Choose Build➪Deploy Solution.**

   Visual Studio deploys the stored procedure to the Adventure Works database.

   Deploying the stored procedure creates the stored procedure in SQL Server.

You can use Server Explorer in Visual Studio or the SQL Server in Management Studio to execute the stored procedure.

Executing CLR routines

You execute your managed CLR routines by using SQL statements. For example, to execute a managed stored procedure, type the following line:

```
EXEC mystoredprocedure
```

To use Server Explorer to execute the `hellosqlserver` stored procedure from the preceding section, follow these steps:

1. **Expand the Stored Procedures folder in Server Explorer.**

   Press Ctrl+Alt+S to open the Server Explorer window if it's closed.

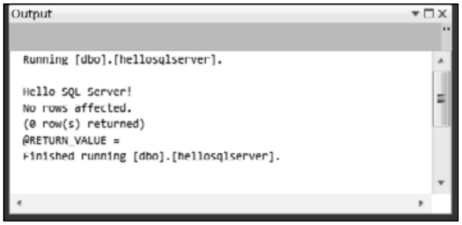
2. **Right-click the `hellosqlserver` stored procedure.**

   You may need to refresh Server Explorer if you don't see the stored procedure. To refresh the Server Explorer contents, right-click the Stored Procedures folder and choose Refresh from the contextual menu.

3. **Choose Execute from the contextual menu.**

   The output from the stored procedure appears in the Output window, as shown in Figure 6-9.

   If an error message tells you to enable CLR integration, see the earlier section "Enabling integration with SQL Server 2008."



```
Running [dbo].[hellosqlserver].

Hello SQL Server!
No rows affected.
(0 row(s) returned)
@RETURN_VALUE =
Finished running [dbo].[hellosqlserver].
```

Figure 6-9: The stored procedure's output appears in the output window.

Debugging SQL Server projects

Visual Studio provides support for debugging SQL Server projects. You must enable debugging before you can use the debugger. To enable debugging, right-click the database connection in Server Explorer and choose Allow SQL/CLR Debugging.

To debug the `hellosqlserver` stored procedure, follow these steps:

1. **Right-click the Test Scripts folder in Solution Explorer.**
2. **Choose Add Test Script from the contextual menu.**

   A new test script appears in the Code Editor.
3. **Type** exec hellosqlserver **in the script.**
4. **Insert a breakpoint by clicking the gray border to the right of the statement you type in Step 3.**

   A red dot appears next to the statement.
5. **Right-click the test script in Solution Explorer and choose Set As Default Debug Script from the contextual menu.**
6. **Press F5 to start debugging.**

   The debugger stops at your breakpoint.
7. **Press F11 to step through each line in your stored procedure.**

To read more about using the debugger, see Book III, Chapter 7.

You can also start the debugger by right-clicking the stored procedure in Server Explorer and choosing Step into Stored Procedure.

Managing Your Database with Visual Database Tools

Visual Studio provides an extensive set of tools for creating and managing database objects. Almost any time you work with database objects or SQL Server projects, you're using one of the Visual Database Tools. Although the tools work with many kinds of databases, they're tightly integrated with SQL Server.

Visual Studio includes these tools for manipulating databases:

- **Server Explorer:** Connects to databases and accesses database objects
- **Database Diagram Designer:** Views or modifies a database's structure using a visual designer
- **Table Designer:** Creates and modifies tables
- **Query and View Designer:** Creates views and queries

Server Explorer is the key to accessing Visual Database Tools. Database objects appear in a hierarchical tree view in Server Explorer. You invoke most tools by right-clicking a database object in Server Explorer and choosing a tool from the contextual menu.

Change the hierarchical view of Server Explorer by right-clicking a data connection and choosing Change View from the contextual menu.

Before you can work with a database in Server Explorer, you must create a connection to a database. For more information, see the section about connecting to a database in Chapter 2 of Book V.

You can use Server Explorer to create a new database in SQL Server. To create a new database, follow these steps:

1. **Choose View⇨Server Explorer to open Server Explorer.**
2. **Right-click the Data Connections folder.**
3. **Choose Create New SQL Server Database from the contextual menu.**

   The Create New SQL Server Database dialog box appears.
4. **Type your server's name or select a server from the Server Name drop-down list.**

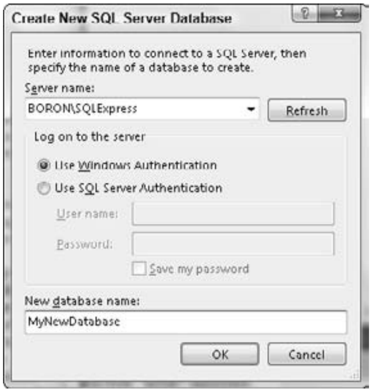
   Be sure to include the instance name. The default instance name for SQL Server 2008 Express is SQLExpress. The default instance name for SQL Server 2008 Developer Edition is MSSQLSERVER.

5. **Select the appropriate authentication type and enter your login credentials, if required.**

6. **Type a name for the database, such as** MyNewDatabase**, in the New Database Name field, as shown in Figure 6-10.**

7. **Click OK.**

   A connection to the new database appears in Server Explorer.



Create New SQL Server Database

Enter information to connect to a SQL Server, then specify the name of a database to create.

Server name: BORON\SQLExpress   Refresh

Log on to the server

● Use Windows Authentication

○ Use SQL Server Authentication

User name:

Password:

□ Save my password

New database name: MyNewDatabase

OK   Cancel

Figure 6-10: Type a name for the new database.

Using a local database

In most cases, you create a database on a database server running SQL Server. Sometimes, however, you may wish to run the database on the local computer. A database that you deploy with your application is called a *local database.* You may choose to store your application's data in a local database rather than an XML file or a plain text file. Visual Studio supports using SQL Server and SQL Server Express `.mdf` database files and Access `.mdb` database files as local databases. All the Visual Database Tools work with local databases.

Here's how to create a new local database:

1. **Create a new Windows application.**

   See Book IV, Chapter 1 for more information on creating a Windows application.

2. **Right-click the project folder in Solution Explorer.**

3. **Choose Add➪New Item from the contextual menu.**

   The Add New Item window appears.

4. **Click the SQL Database icon.**
5. **Type a name for the database in the Name text box and then click the Add button.**

   The database file is added to Solution Explorer. A data connection to the local database file appears in Server Explorer. The Data Source Configuration Wizard starts.
6. **Click Finish to exit the wizard.**

The Data Source Configuration Wizard creates a strongly typed `DataSet`. You use the Visual Database Tools to add tables, queries, and views to your local database, as described in the next two sections. You access local databases using ADO.NET the same way you access remote databases.

You may also add an existing SQL Server or Access database to your project. To add an existing database file, right-click the project folder and choose Add➪Existing Item. You need to create a database connection to your local file in order to access the database. See Book V, Chapter 2 to read about connecting to databases.

TIP

Follow Steps 2 through 5 of the preceding list to add a local database to a Web site. The Data Source Configuration Wizard doesn't start. The default application services provider database used for Membership and other ASP. NET services is a local database.

Deploying a local database

When you deploy a project that uses a local database, Visual Studio automatically includes SQL Server 2008 Express Edition. If the local client that installs your application doesn't already have SQL Server installed, it's installed with your application.

The first time you build your application, Visual Studio places a copy of the database file into your application's bin folder with all the rest of the application's output, such as executable and DLL files. You have three choices for how Visual Studio manages the database file each time you build:

- **Copy If Newer:** Copies the database file from your project to the bin folder if the project's file has been updated more recently than the copy in the bin folder.
- **Copy Always:** Copies the file from your project to the bin folder every time. This is the default action.
- **Do Not Copy:** Never copies the file from your project to the bin folder.

You have to decide which approach is the most appropriate for your situation. Usually, the Copy Always option works fine.

You set the copy option using the `Copy to Output Directory` property on the local database file. Access the property using the Properties window by right-clicking the local database file and choosing Properties.

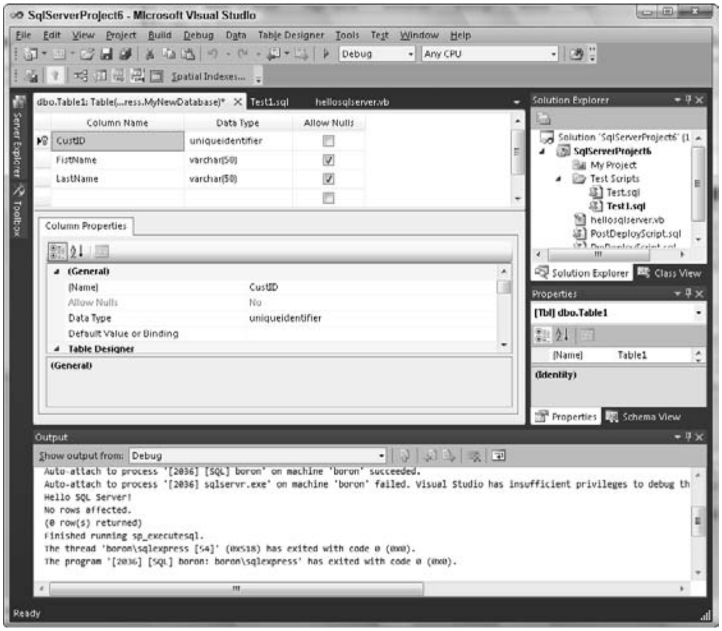
Adding tables with Table Designer

Add tables to a database by using Table Designer. To add a new table to the database you create in the preceding section, follow these steps:

1. **Click the plus sign (+) to expand the data connection in Server Explorer.**
2. **Right-click the Tables folder.**
3. **Choose Add New Table from the contextual menu.**

   Table Designer displays a grid that you use to add new columns.
4. **Add these columns to the table by using the grid:**
   - CustID, data type `uniqueidentifier`
   - FirstName, data type `varchar`
   - LastName, data type `varchar`
5. **Right-click the CustID column.**
6. **Choose Set Primary Key from the contextual menu.**

   A key appears next to the column name, as shown in Figure 6-11.
7. **Press F4 to display the table's Properties window.**



	Column Name	Data Type	Allow Nulls
▶🔑	CustID	uniqueidentifier	□
	FirstName	varchar(50)	☑
	LastName	varchar(50)	☑
			□

Column Properties

(General)
(Name) CustID
Allow Nulls No
Data Type uniqueidentifier
Default Value or Binding
Table Designer

(General)

[Tbl] dbo.Table1
(Name) Table1
(Identity)

Show output from: Debug

```
Auto-attach to process '[2036] [SQL] boron' on machine 'boron' succeeded.
Auto-attach to process '[2036] sqlservr.exe' on machine 'boron' failed. Visual Studio has insufficient privileges to debug th
Hello SQL Server!
No rows affected.
(0 row(s) returned)
Finished running sp_executesql.
The thread 'boron\sqlexpress [54]' (0x518) has exited with code 0 (0x0).
The program '[2036] [SQL] boron: boron\sqlexpress' has exited with code 0 (0x0).
```

Figure 6-11: A key appears near to the column name.

8. **Set the `Name` property to Customer.**

9. **Save the table.**

   Visual Studio adds the table to the database, and it appears in Server Explorer.

Table Designer is capable of doing much more than just adding columns. Click the Table Designer menu to see a list of commands that enable you to

- Create foreign key relationships.
- Create indexes.
- Add check constraints.

Table Designer generates change scripts that execute the changes you make in the designer in SQL Server. You can harness the power of these scripts and reuse them, modify them, or execute them later.

To generate change scripts, follow these steps:

1. **Create a new table or modify an existing table in Table Designer.**

   Double-click an existing table object in Server Explorer to open it in Table Designer.

2. **Choose Table Designer➪Generate Change Script.**

   The Save Change Script dialog box appears.

3. **Click Yes to save the script.**

TIP

To generate create scripts in database projects, right-click the table in Server Explorer and choose Generate Create Script to Project from the contextual menu. In a database project, you can generate create scripts for any table in Server Explorer. Create scripts are most often used to add tables or fields to a database, and on rare occasions to add databases themselves.

Adding queries and views

Query and View Designer is a visual tool for creating queries and views. The tool is known as Query Designer when it's used for creating queries and View Designer when used to create views.

REMEMBER

Query Designer is also known as *Query Builder.*

Use a query any time you need to execute an SQL statement against the database. Create a view when you want to save your query in the database.

You can save queries in a database project.

Examples of queries you can create with Query and View Designer include

- **Select:** Retrieves rows by using a `SELECT` statement
- **Insert Results:** Copies rows from one table to another by using an `INSERT INTO ... SELECT` statement
- **Insert Values:** Inserts a new row by using the `INSERT INTO ... VALUES` statement
- **Update:** Updates rows by using an `UPDATE` statement
- **Delete:** Deletes rows by using the `DELETE` statement
- **Make Table:** Copies results of a query into a new table by using the `SELECT ... INTO` statement

To start Query Designer, follow these steps:

1. **Right-click an existing data connection in Server Explorer.**
2. **Choose New Query from the contextual menu.**

   The Query Designer opens.

To start View Designer, follow these steps:

1. **Right-click the Views folder in an existing data connection in Server Explorer.**
2. **Choose Add New View from the contextual menu.**

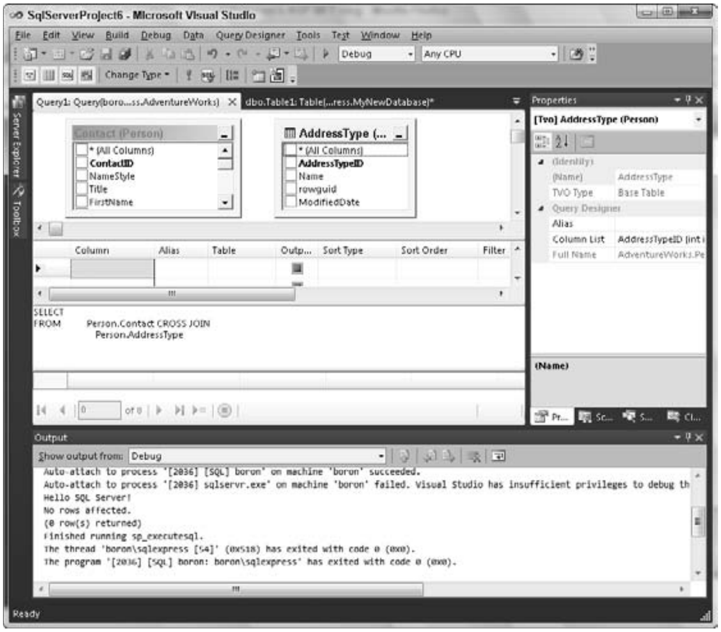
   The View Designer opens.

Query and View Designer has three panes that help you build SQL statements. Figure 6-12 shows Query Designer.

The three panes, from top to bottom, are

- **Grid:** Specifies which columns to display and how to sort, group by, and filter
- **SQL:** Displays the SQL created by the Designer
- **Results:** Displays the query's output

To see the Query Designer in action, check out the section about updating data with the data designer in Chapter 2 of Book V.



```
SELECT
FROM        Person.Contact CROSS JOIN
              Person.AddressType
```

[Tvo] AddressType (Person)

- (Identity)
  - (Name): AddressType
  - TVO Type: Base Table
- Query Designer
  - Alias:
  - Column List: AddressTypeID (int i
  - Full Name: AdventureWorks.Pe

(Name)

Show output from: Debug

```
Auto-attach to process '[2036] [SQL] boron' on machine 'boron' succeeded.
Auto-attach to process '[2036] sqlservr.exe' on machine 'boron' failed. Visual Studio has insufficient privileges to debug th
Hello SQL Server!
No rows affected.
(0 row(s) returned)
Finished running sp_executesql.
The thread 'boron\sqlexpress [54]' (0x518) has exited with code 0 (0x0).
The program '[2036] [SQL] boron: boron\sqlexpress' has exited with code 0 (0x0).
```

Figure 6-12: Query and View Designer has three panes.

Chapter 7: LINQ

In This Chapter

- **Seeing how LINQ differs from SQL**
- **Creating a simple program to query integers**
- **Finding active processes**
- **Creating new object queries**

Earlier chapters of this book use SQL Server as a data store and then use the objects that Visual Studio makes available to access the data in the databases. And although that's a large portion of data access techniques, there's more to managing data than database storage. With .NET 3.0, Microsoft took data access to the next level by integrating it into the .NET programming languages. This cool new feature is called Language Integrated Query (LINQ), and it allows you to create powerful queries on not only database storage, but on any collection of data in your programs. Seemingly, every Web site on .NET technologies is talking about it.

LINQ, which is integrated into the newest versions of C# and VB.NET, helps you organize your data structures in a manner similar to how databases organize data. Similar to SQL, you can query program data. For example, say that you have a list of data structures, and you want to find the one that has an `ID` of 26. You can use LINQ to query the data and find the structure you're looking for.

LINQ isn't just an enhancement to the collection classes. Instead, it's built right into the language. Yup, the querying capabilities are built right into the language. You can run queries against pretty much any kind of data — not just collections. This is one of those language features that you'll wonder how you got along without. LINQ is a set of extensions to the .NET Framework that encompasses language-integrated query, set, and transform operations. It extends C# and Visual Basic with native language syntax for queries and provides class libraries to take advantage of these capabilities.

Using LINQ to Query Integers

LINQ is so simple that the best way to see how to use it is to dive in and get going. The first program shown here illustrates how to create an array of integers that you'll initialize in a loop. Then you can see how to use LINQ to query the array for integers greater than 55.

First, write the code that initializes an integer array from which you'll obtain the query results. The VB and C# code follows:

VB:

```
Dim nValues(50) As Integer
For i = 0 To 49
     nValues(i) = 1 + i * 2
Next
```

C#:

```
int[] nValues = new int[50];
for (int i = 0; i < 50; i++)
{
     nValues[i] = 1 + i * 2;
}
```

The next thing to do is perform the query. LINQ queries resemble SQL queries. They have a `from` clause, a `where` clause, and a `select` clause. LINQ queries are structured slightly differently than SQL queries, though. The following is a LINQ query that obtains all numbers greater than 55 from the integer array created from the previous code snippet:

VB:

```
Dim Results = From n In nValues Where n > 55 Select n
```

C#:

```
var Results = from n in nValues where n > 55 select n;
```

Figure 7-1 shows the relationship of an SQL query to the equivalent LINQ query.

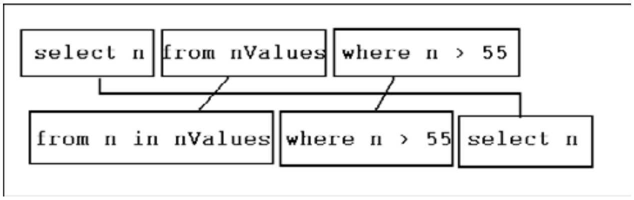


Figure 7-1: The LINQ query has many of the same elements of an SQL query, but they are structured slightly differently.

You can see the results of the running program in Figure 7-2.

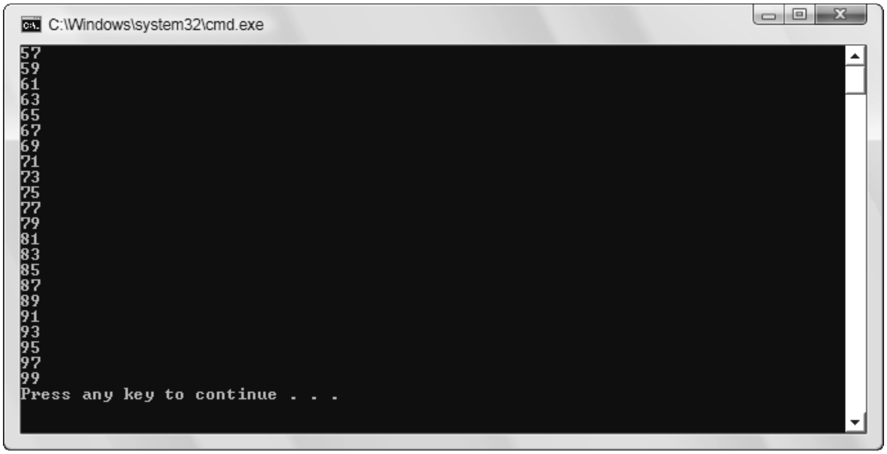
The complete program in VB and C# follows:

VB:

```
Sub Main()
     Dim nValues(50) As Integer
     For i = 0 To 49
          nValues(i) = 1 + i * 2
     Next

     Dim Results = From n In nValues Where n > 55 Select n

     For Each Result In Results
          Console.WriteLine(Result)
     Next

End Sub
```

C#:

```
static void Main(string[] args)
{
     int[] nValues = new int[50];
     for (int i = 0; i < 50; i++)
     {
          nValues[i] = 1 + i * 2;
     }

     var Results = from n in nValues where n > 55 select n;

     foreach (var Result in Results)
     {
          Console.WriteLine(Result);
     }

}
```



```
57
59
61
63
65
67
69
71
73
75
77
79
81
83
85
87
89
91
93
95
97
99
Press any key to continue . . .
```

Figure 7-2: This console application shows all numbers, from the list, which are greater than 55.

You can download the complete program (both C# and VB versions) for this example from this book's companion Web site at `www.dummies.com/go/vs2010`.

In the `Linq1` program in the preceding example, notice the data type of the `Results` variable: There isn't one. Instead, note the datatype called `var`, which means the compiler will figure out (at compile time) what type to make for the variable. When programming with LINQ, this helps simplify matters.

Finding Active Processes

In this example, you can see how to create another console application that finds the current active processes.

You need the `System.Diagnostics` namespace for this program.

The query for this is as follows:

VB:

```
Dim Processes = From p In Process.GetProcesses() Select p
```

C#:

```
var Processes = from p in Process.GetProcesses() select p;
```

There is no `where` clause. Like with SQL queries, `where` clauses are optional.

You could add a `where` clause that will find only process names that contain the letter *x,* as follows:

VB:

```
Dim Processes = From p In Process.GetProcesses() _
           Where p.ProcessName.Contains("x") Select p
```

C#:

```
var Processes = from p in Process.GetProcesses()
    where p.ProcessName.Contains("x") select p;
```

You can download the complete program (both C# and VB versions) for this example from this book's companion Web site at `www.dummies.com/go/vs2010`.

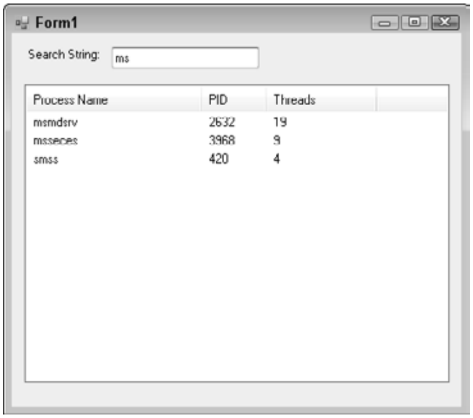
Notice how the query is set up: A from clause specifies the collection of process names returned by Process.GetProcesses. Then, a where clause gives the criteria. Concluding is the select clause.

Visual Studio's IntelliSense (that thing that pops up and shows you the members of your variables, for example) is incredibly smart. When you start typing a where clause, IntelliSense kicks in. But look closely: IntelliSense knows the type of the var in the expression (p, in the preceding example). In order to determine the type of p, IntelliSense had to

1. **Parse the second half of the from clause.**
2. **Determine that Process.GetProcesses returns a collection of Process types.**
3. **Knowing that p is an individual element in the collection, determine that p's type is Process.**

Whew!

To finish off this section, I created a program with a graphical user interface (GUI). You can enter a search string, and it will match your search string to process names that are active. You can see the program running in Figure 7-3.



Search String: ms

Process Name	PID	Threads
msmdsrv	2632	19
mssecces	3968	9
smss	420	4

Figure 7-3: This program has a GUI and allows users to enter a search string.

You can download the complete program (both C# and VB versions) for this example from this book's companion Web site at www.dummies.com/go/vs2010.

Creating New Objects in the Query

The select clause may seem unnecessary at first. In the previous examples, the select p business would probably seem obvious. Seemingly, the code

```
from n in numbers where n > 25
```

should be all you need. However, the select clause does serve a purpose; in it, you can create new objects based on those objects that fit the criteria in the where clause.

Have a look at this code, which is a slightly modified version of the previous:

```
static void Main(string[] args)
{
     var Processes = from p in Process.GetProcesses()
          where p.ProcessName.Contains("x")
          select new { p.ProcessName, p.Id};

     foreach( var Proc in Processes )
     {
          Console.WriteLine( Proc.ProcessName );
          Console.WriteLine( Proc.Id );
     }
}
```

Look carefully at the select clause. It creates a set of new objects, each with a ProcessName member and an Id member. These new objects don't have a class name; you're creating an anonymous type. The new statement is called for each item in the resulting set, and the members of the anonymous object are initialized with the values in p.ProcessName and p.Id. (The names of the members of the anonymous class are inherited from the original object; thus, the names are ProcessName and Id.)

Now, look at the foreach statement. The members of the collection are no longer instances of Process, so you can't declare the loop variable as Process. Instead, just use var to let the compiler figure it out for you.

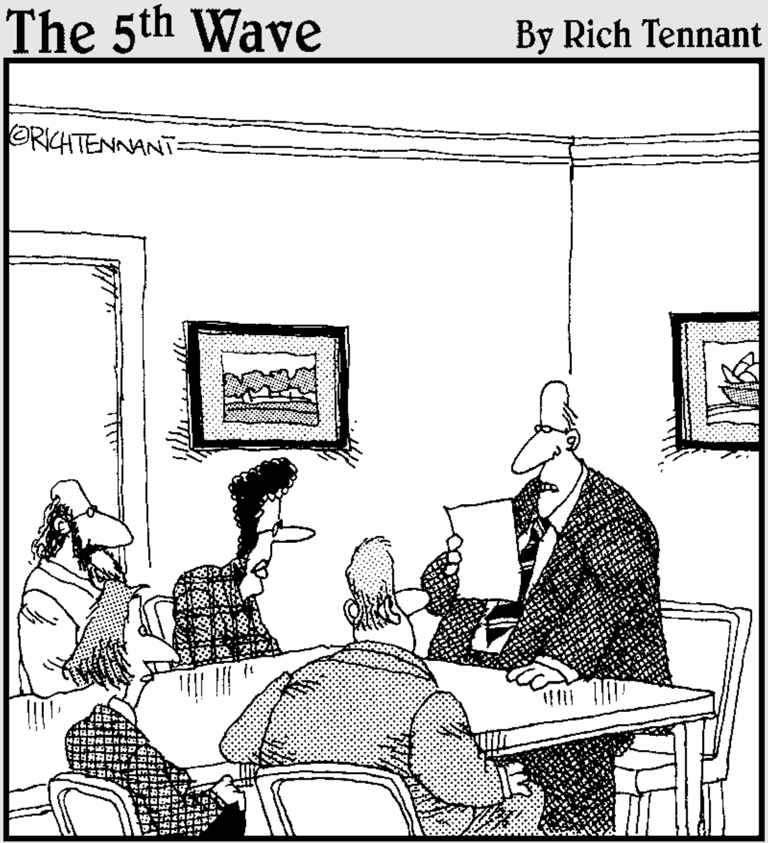
The type is anonymous, so it doesn't have a direct name.

The anonymous types you create actually have kind of a cool feature: They automatically get a ToString() method. You could code your foreach statement like so:

```
foreach (var x in processes)
{
    Console.WriteLine(x.ToString());
}
```

Book VI

Going the Extra Mile



"We're much better prepared for this upgrade than before. We're giving users additional training, better manuals, and a morphine drip."

Contents at a Glance

Chapter 1: Building Solutions and Projects

In This Chapter

- **Understanding the build process**
- **Creating build configurations**
- **Viewing build feedback**
- **Automating daily build**

An application comprises many pieces — presentation code, data-access libraries, unit tests, configuration files, installation programs, and more. Visual Studio organizes these pieces into *projects,* and the entire application is a *solution.*

While you're creating the interfaces and writing the data access code for your solution, you need to stop periodically and test your application. To test your application, you need to run a compiler that converts the human-readable source code in your projects to machine-readable binary files. Visual Studio provides a set of build commands that you use to compile your projects and solutions.

By its nature, the build process is tightly coupled with deployment. In the simplest deployment case, you create a release build and distribute the output files. However, very rarely is real life so simple. Some types of applications — such as multiple-tier Windows applications, ASP.NET Web sites, and mobile applications — are inherently more difficult to deploy. You need to take many variables into consideration when creating release builds.

This chapter shows you how to build your solutions and projects for use on a local development computer. (See Chapter 2 in Book VI to read about deploying release builds.)

Understanding Solutions and Projects

Visual Studio uses logical containers — *projects* and *solutions* — for grouping all the code and resources associated with an application.

- *Solutions* contain all the projects necessary to build an application.
- *Projects* include all the source code and support files necessary to create the application.

For example, a solution may contain one project each for the following:

- Windows Forms
- Data-access code
- Database code
- Testing code
- The installer

The process of converting all the code found in a set of projects to an application is *building* or *creating a build.* Builds have settings that define where the output files are located and whether debug information is included. A group of settings is a *build configuration.* Visual Studio solutions and projects include two default build configurations:

- **Debug:** Code compiles with information for debugging and no optimization. The nature of the build is to enable you to test your code.
- **Release:** Code compiles with optimizations and no debugging information. The release configuration is intended for production deployment.

Builds can target a specific hardware platform, such as the 32-bit (x86) or 64-bit (x64) architectures. The Debug and Release build configurations provided by Visual Studio target the x86 architecture.

The anatomy of a build

Building software takes a set of inputs and creates a set of outputs. The inputs include the following:

- Source-code files
- Resource files
- Configuration files
- Local database files

The build process takes these files and converts them to a set of outputs that include

- **Tokenized files,** such as application executables and dynamic link libraries (DLLs). Binary files are *assemblies.*

- **Debugging information,** which is stored in a program database (PDB) file. (See Book III, Chapter 7 for more information about debugging.)
- **Feedback messages,** such as compiler warnings and errors.

The build process uses a compiler to convert your human-readable input files into machine-readable binary files. Each programming language uses its own compiler.

You use the outputs from a build to accomplish the following:

- **Test an application you're developing.**
- **Deploy an application to a test or production environment.**
- **Determine whether you can check in your source code to form part of your source-code repository.**

  Inputs to the build process — the source code — are *checked in* (established as source code) after you have a successful build.

Figure 1-1 shows the build process. Note that the outputs are deployed, but usually they're not checked into source code control.

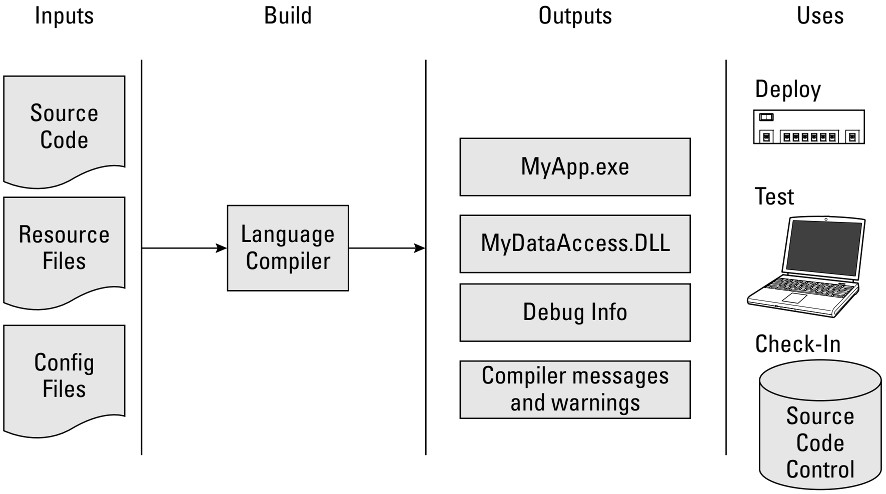


Figure 1-1: The build process creates a set of deployable outputs.

Using the Build menu

Visual Studio provides a Build menu from which you can launch different build commands for your solution or project. The options on the Build menu are

- **Build Solution:** Compiles any source code that hasn't yet been compiled. The Build command doesn't delete any existing output files. Instead, the Build command performs an incremental build of any files not yet compiled.
- **Rebuild Solution:** Deletes, or *cleans,* any existing output files and compiles all source code into binary files.
- **Clean Solution:** Deletes all the output files generated by previous builds.
- **Publish:** Calls the Publishing Wizard to build and deploy the solution. (See Chapter 2 in Book VI for more information on using the Publishing Wizard.)
- **Batch Build:** Compiles multiple build configurations as a batch.

Deployable projects, such as SQL Server Reporting Projects, display the Deploy command on the Build menu. If you have the Web Deployment Project Add-In installed, you also have the option Add Web Deployment Project on the Build menu. (See Chapter 2 in Book VI for more information about Web Deployment Projects.)

You execute build commands against solutions or individual projects. Here are some scenarios for how you can use the Build menu:

- **Build projects individually when a solution has many projects.** Compiling large solutions and projects is time consuming. For that reason, you may choose to build individual projects as you need them rather than building the entire solution. Using the Build command to incrementally compile source files can also save time.
- **Build each project in a solution individually when a dependent project generates compile errors.** When a solution has many projects that are dependent upon one another, an error in one project cascades to the dependent projects, which makes finding the original error difficult. Instead of plowing through a long list of errors and warnings, compile each project individually and deal with any errors you find.
- **Build an entire solution.** If your solution builds lightning-fast, you can use either the Build or the Rebuild Solution command.

  Use the Rebuild Solution option (rather than Build) if you're getting strange errors that you don't think you should be getting. Using Rebuild Solution wipes out all the previous build's outputs.
- **Build individual projects as changes are made to them.** When you have large projects and solutions, building is faster than rebuilding the entire solution.

Using the Start command from the Debug menu calls the Build command and executes your application. During development, many developers press the F5 key with the expectation that their code is going to build and execute.

Only after they see that long list of compiler errors do they start using the Build menu.

The commands on the Build menu correspond to targets files used by the Microsoft Build Engine (MSBuild). To read more about MSBuild, see the upcoming section “Automating builds.”

Selecting the active build and platform

When you build a solution or a project, you select a named build configuration you want to build, such as Debug or Release. Each build configuration has specific properties that determine how the project is built. Visual Studio 2010 automatically creates the Debug and Release build configurations for each project and solution you create.

The build process creates code for a specific hardware platform (such as the Intel x86 or x64 architectures). You pick a hardware platform when you build your solution or project.

To set a build configuration and a hardware platform for a build, use the Solution Configuration and Solution Platform drop-down lists on the Standard toolbar, as shown in Figure 1-2.

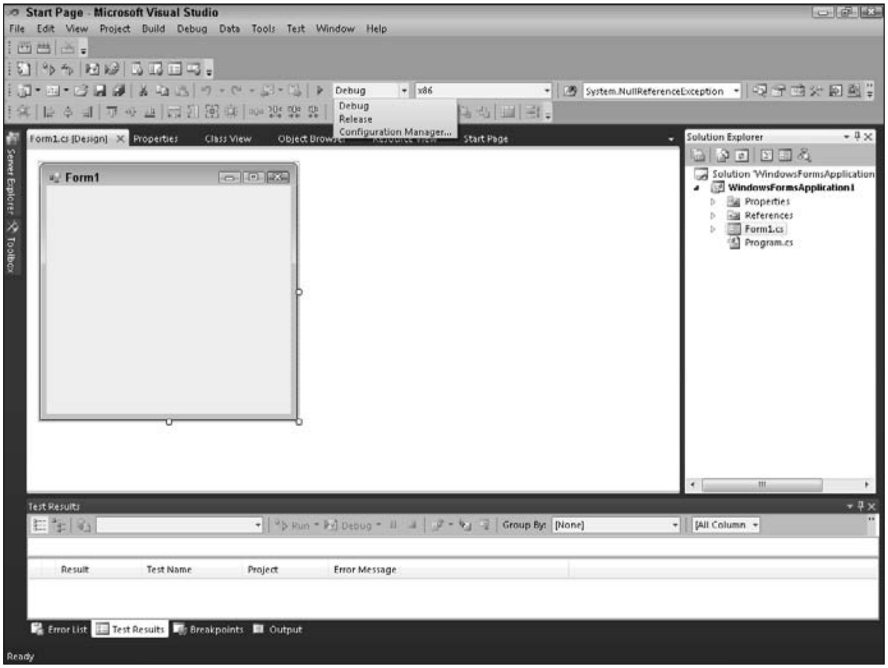


Figure 1-2: Select the build type.

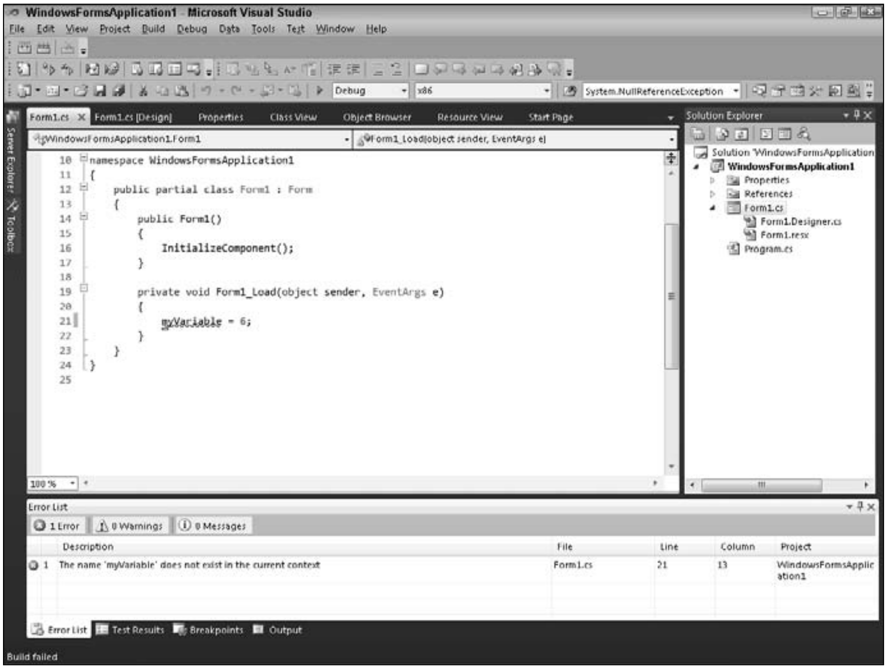
Dealing with compiler errors and output

The major function of the build process is to compile your source code into binary files. The build process provides output that shows the list of commands executed, along with any feedback from those commands. If the compiler can't compile your code, error messages appear in the Error List window.

If you execute your build by using the Start command from the Debug menu, a dialog box warns you that the build was unsuccessful. If you use the Build menu, the Error List window appears. Figure 1-3 shows an error message that cropped up when trying to use a variable that hadn't yet been declared.

Each programming language uses its own compiler. The message displayed in Figure 1-3 is from the C# compiler. The Visual Basic compiler displays the following message for the same error:

```
Name 'myVariable' is not declared.
```



```
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            myVariable = 6;
        }
    }
}
```

	Description	File	Line	Column	Project
1	The name 'myVariable' does not exist in the current context	Form1.cs	21	13	WindowsFormsApplication1

Figure 1-3: Error and warning messages from the compiler appear here.

Among other things, you can use the Errors window to

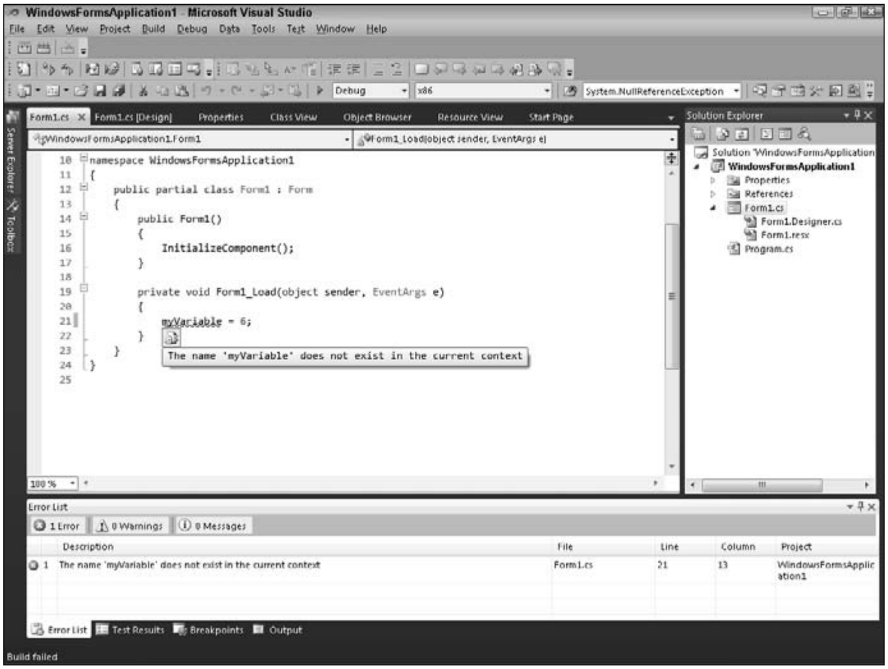
- **Go to the source code where the error occurred.** When you double-click an error, Visual Studio opens the source-code file and positions the cursor where the error occurred.
- **Get help on an error.** Right-click an error and choose Show Error Help from the shortcut menu.

Hover your mouse pointer over an error in source code to display the error in a ToolTip, as shown in Figure 1-4.

Use the Output window to view the commands executed by the build process. The Output window displays the following information:

- Build configuration
- Compiler and switches used
- Output from the compiler
- Build status

Messages appear for each project compiled in the build. Figure 1-5 shows a successful build for a single project, using the Debug build configuration.



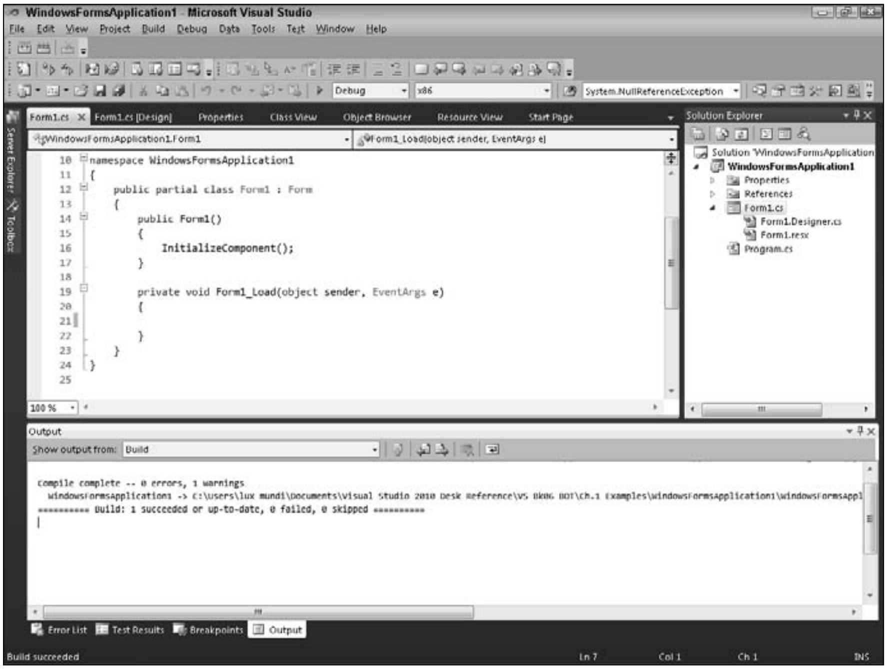
```
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            myVariable = 6;
        }
    }
}
```

The name 'myVariable' does not exist in the current context

	Description	File	Line	Column	Project
1	The name 'myVariable' does not exist in the current context	Form1.cs	21	13	WindowsFormsApplication1

Figure 1-4: The ToolTip shows the error.

The long line that appears in Figure 1-5 between the first and last lines is a single command that executes the compiler. To create this command, the Build command pulls all the project settings, references, and properties that you set in Visual Studio.



```
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
}
```

```
Compile complete -- 0 errors, 1 warnings
  WindowsFormsApplication1 -> C:\Users\lux mundi\Documents\Visual Studio 2010 Desk Reference\VS 8K06 DOT\Ch.1 Examples\WindowsFormsApplication1\WindowsFormsAppl
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Figure 1-5: The Output window displays the compiler command used to build your project.

You can copy and paste the compiler command from the Output window to reuse on the command line or in a batch file. Batch files are a good way to automate builds. You'll probably need to qualify the path names to your project's resources. To open a Visual Studio command prompt, choose Start➪All Programs➪Microsoft Visual Studio 2010➪Visual Studio Tools➪Visual Studio Command Prompt (2010).

To specify whether a file should be compiled with a project, use the Properties window to set the file's `BuildAction` property.

Using Configuration Manager

You use Configuration Manager to create new build configurations and manage existing configurations for solutions and projects. You use Project Designer to associate a project's properties with its build configuration.

To create a new build configuration by using Configuration Manager, follow these steps:

1. **Choose Configuration Manager from the Build drop-down list.**

   Configuration Manager appears, displaying the configuration settings for the configuration and platform selected at the top.

2. **Choose New from the Active solution configuration drop-down list.**

   The New Solution Configuration window appears.

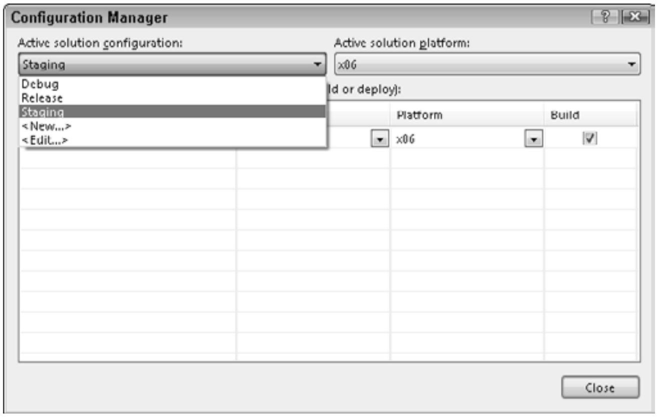
3. **Type** Staging **in the Name field.**

4. **Choose Debug from the Copy Settings From drop-down list.**

5. **Leave the Create New Project Configurations check box selected.**

6. **Click OK.**

   The new configuration appears in Configuration Manager, as shown in Figure 1-6.



Configuration Manager

Active solution configuration: Staging

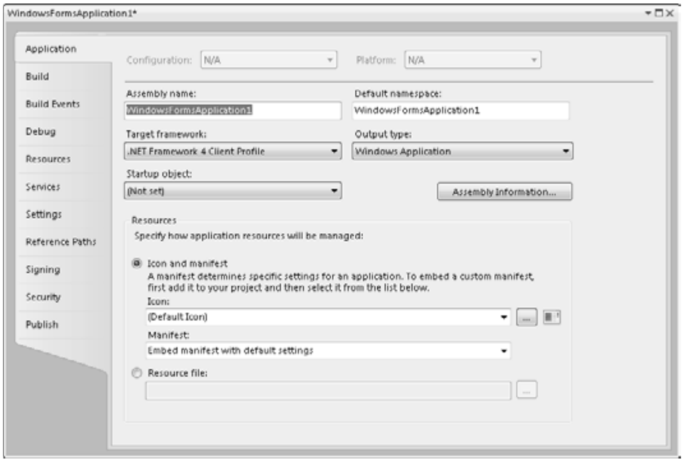
Active solution platform: x06

Figure 1-6: New build configurations appear in Configuration Manager.

A solution's build configuration is composed of two main elements:

- **Project configurations:** A set of build-and-debug properties defined for a given project.
- **Platforms:** A setting that specifies what hardware platform the build targets.

For each project in the solution, you select a project configuration and a platform, as shown in Figure 1-6. You create new project configurations and hardware platforms, if necessary. You include the project in the build by placing a check in the Build column. You use Project Designer to set the properties for each project configuration. Figure 1-7 shows an example of selecting a project configuration for a solution configuration, which you can get to by going to the project properties under the Project menu.



Configuration: N/A
Platform: N/A

Assembly name: WindowsFormsApplication1
Default namespace: WindowsFormsApplication1

Target framework: .NET Framework 4 Client Profile
Output type: Windows Application

Startup object: (Not set)

Resources
Specify how application resources will be managed:

● Icon and manifest
A manifest determines specific settings for an application. To embed a custom manifest, first add it to your project and then select it from the list below.
Icon: (Default Icon)
Manifest: Embed manifest with default settings

○ Resource file:

Figure 1-7: You can set most of the project configur-ation in Project Designer.

Visual Studio writes the build configurations you create in Configuration Manager to the appropriate solution or project file. When you build your solution or project, MSBuild uses the solution or project files as a script. For example, the following line is added to the solution file for the staging build configuration:

```
Staging|Any CPU = Staging|Any CPU
```

Open your solution file or project file in any text editor to view the changes made by Configuration Manager. (See the section “Automating builds,” later in this chapter, to read more about the Microsoft Build Engine.)

If you have a specific hardware platform in mind, you can use Configuration Manager to specify that a build should target that platform. For example, to create a new Debug solution configuration that targets the x64 platform, follow these steps:

1. **Open Configuration Manager.**
2. **Choose Debug from the Active Solution Configuration drop-down list.**
3. **Choose New from the Active Solution Platform drop-down list.**

   The New Solution Platform window appears.
4. **Select x64 as the new hardware platform.**
5. **Click OK.**

   A new Debug solution configuration for the x64 platform is created.

If you're using the MSBuild Toolkit, you can select a version of the .NET Framework to target using the Platform. (See Book II, Chapter 4 for more information about using the MSBuild Toolkit to target a specific version of the .NET Framework.)

Setting project configurations

Visual Studio solutions and projects have many settings — *properties* — that define the solution or project. These properties define attributes, such as

- Other code referenced by the project
- Where to place the project's compiled output
- Which debugger to use

You access the properties for a solution or project by choosing Properties from the solution's or project's shortcut menu. The two views for properties are

- **Project Designer:** Displays properties for a project in Document Explorer. All Visual Studio projects use Project Designer to manage their properties.
- **Property Pages:** Displays properties in a dialog box. Solutions and ASP.NET Web sites use the Property Pages window to manage their properties.

ASP.NET Web sites don't use projects, so they don't use Project Designer. Microsoft has a project type (Web Application Projects) that allows you to have ASP.NET projects.

Using Project Designer

Visual Studio 2010 includes a new feature — Project Designer — that you use to access your project's configuration settings in a single designer.

To open Project Designer, follow these steps:

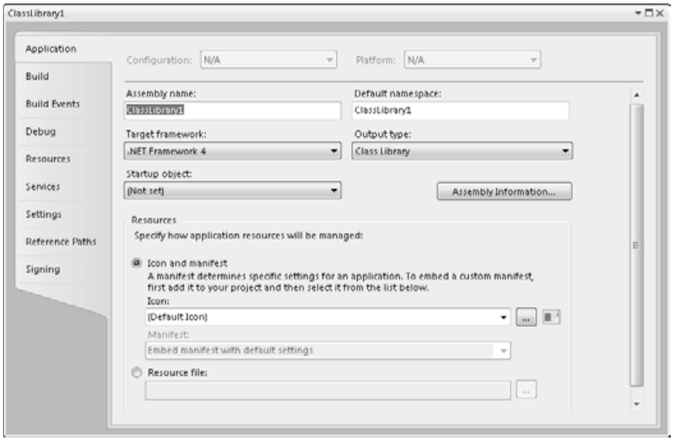
1. **Right-click your project in Solution Explorer.**

   A shortcut menu appears.

2. **Choose Properties.**

   Project Designer appears in Document Explorer, showing a set of tabs along the left that group common properties. To access a set of properties, click its tab. Figure 1-8 shows the Project Designer for a class library project type.

You can also open Project Designer from the Project menu.



Configuration: N/A
Platform: N/A

Assembly name: ClassLibrary1
Default namespace: ClassLibrary1

Target framework: .NET Framework 4
Output type: Class Library

Startup object: (Not set)

Resources

Specify how application resources will be managed:

● Icon and manifest

A manifest determines specific settings for an application. To embed a custom manifest, first add it to your project and then select it from the list below.

Icon: (Default Icon)

Manifest: Embed manifest with default settings

○ Resource file:

Figure 1-8: This is Project Designer for a class library project.

Some tabs in Project Designer feature properties that you can configure to match your build configuration type. For example, you can set build properties for a Debug build. To set the build properties for a C# project type, follow these steps:

1. **Open Project Designer.**
2. **Click the Build tab.**

   The project's build properties appear.

3. **Choose a build configuration from the Configuration drop-down list.**

   The properties for the selected build configuration appear. If you want to manage properties for all your build configurations, you can choose All Configurations.

4. **Choose a build platform from the Platform drop-down list.**

5. **Set properties in Project Designer.**

   The properties that you choose depend largely on whether you intend to debug or release the output. The properties for Visual Basic projects are slightly different because the compilers are different.

You won't find an OK or a Save button for Project Designer. The properties are saved immediately.

The Project Designer for Visual Basic project types has a Compile tab instead of a Build tab. You can access build properties from the Compile tab.

You can set unique build configuration settings for properties in the Build and Debug tabs in C# project types. For Visual Basic project types, you set these items on the Build and Compile tabs.

Using property pages

Solutions and ASP.NET Web sites use the Property Pages dialog box to manage properties. Here's how to use a solution's property pages to set project dependencies:

1. **Right-click the solution in Solution Explorer.**

   A shortcut menu appears.

2. **Expand Common Properties.**

   A list of properties appears.

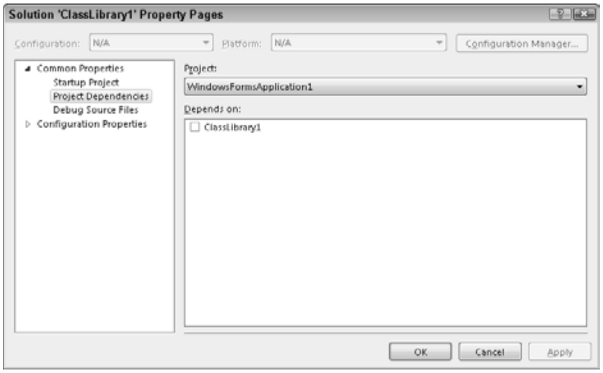
3. **Click Project Dependencies.**

4. **Choose a project from the Project drop-down list (see Figure 1-9).**

   Your project must have more than one project to set project dependencies.

5. **Check the project dependencies and click OK.**

6. **To verify that your projects build in the correct order, choose Projects➪Project Build Order.**



Project: WindowsFormsApplication1

Depends on:

☐ ClassLibrary1

Figure 1-9: Set project dependencies here.

Managing Your Builds

Building software for your own use on a single computer is fairly straightforward. However, most software is developed in teams. Even if you don't plan to release the build, managing builds while a project is being developed is a challenge.

Visual Studio 2010 Professional Edition doesn't provide much support for managing team builds. However, Microsoft offers Team Foundation Server, which targets team-based development. This section discusses some of the best practices for managing builds and then highlights a few tools that can help manage your build process.

Handling Lone Ranger builds

Most builds that occur on an individual developer's workstation are part of the code/build/test cycle. You code a little. You build. You test. You code a little more.

At some point, however, the code must be integrated with the larger code base (*checking in* the new code). Many professionals use source code control software to help manage the code base. Even without source code control, the goal is the same: You need to integrate your changes without breaking your existing code.

Even hobbyists and casual developers can benefit from using source code control. (Chapter 3 in Book VI tells you more about that process and its advantages.)

The integration process for an individual developer, whether that developer is working solo or as part of a team, should go something like this, in this order:

1. **Get the latest source code from the source-code repository before starting any new task.**
2. **Write new code or change existing code.**
3. **Write unit tests for the new code.**
4. **Build the code on the developer's local computer.**
5. **Address any issues with the compiler and run unit tests.**
6. **Get the latest version of source code from the source code repository if other developers are adding code to the repository.**
7. **Build the code again using the updated code from the repository.**
8. **Run unit tests again.**
9. **If the tests pass, check the code into source code control.**

At some point — ideally, daily — a master build should be made from the integrated source code. If you're working alone, you can fire up the master build any time you want. If the master build breaks, the developer who broke the build needs to figure out why — and get the code working again.

Not much time elapses between the point where a new task is started and the new code is integrated into the master source code. Code is integrated when the master source code compiles successfully and passes all unit tests.

Code kept on a developer's local workstation doesn't count for anything. Code isn't useful unless it's integrated.

Creating master builds

The build created from an integrated set of source code is the *master build.* The accepted standard is that master builds should be built daily. Tools are available that create master builds continuously so that developers have instant feedback about whether their code integrates successfully.

Master builds use the latest version of the code from source code control to create the build. Every morning, the first thing a developer does is check the status of the build. If the master build is broken, the developer can't work. Fixing the master build is the number-one priority. The developer responsible for breaking the build is usually the one responsible for fixing it.

If the master build is successful, all developers pull down the latest version of code. Each developer integrates continuously throughout the day.

You may need to distribute the build's output to customers, testers, and developers. Ideally, the master build's output should be kept in a common repository; folks who need access should be able to download from there.

Automating builds

Regardless of how frequently you build your master build, your build process should be automatic and easy to execute. Ideally, you should just have to click one button or execute one command to start the process. For daily builds, you can add your build command to the server's scheduler.

If your build process requires more than just the click of a button, chances are that you won't be able to repeat your build process. Repeatable build processes make it possible to re-create a specific build when beta testers call and complain that their versions aren't working.

Daily builds are usually part of a larger process that may also involve some other tasks:

- Executing automated unit and regression tests
- Running static code analysis
- Creating backups of source code
- Logging the steps of the build
- Sending e-mails to all interested parties when the build is complete

The most important feature of any build is a *basic functional test* (BFT), which is a thorough (and preferably automated) test of your software's major features after the build. You know that you have a good build when you pass the BFT. If your application doesn't pass the BFT, you need to get the responsible developers to debug and rebuild until the build passes.

Your master build process should be separate from Visual Studio, and your build process should be doable in one click. It's not a one-click process if you have to open Visual Studio, open a solution, and then choose the Build command. For master builds, you should use a *build engine,* which is a program that allows you to automate your build process. Two popular build engines are

- **Microsoft Build Engine (MSBuild):** When you use the Build menu in Visual Studio 2010, you're actually sending commands to the Microsoft build engine, MSBuild. MSBuild is a separate tool that specializes in building .NET projects. You can run MSBuild from the command line and build your projects without having Visual Studio installed.

  MSBuild uses XML configuration files to define how projects are built. Visual Studio project files include the XML syntax that MSBuild uses to build the project. You can create your own MSBuild XML configuration files, which is helpful if you want to completely automate your build and run it on a dedicated build machine.

- **NAnt:** This is a popular open-source tool for creating builds from the command line. Like MSBuild, NAnt uses XML files to script the build process. NAnt built files end with the file extension `.build`.

  You can download NAnt for free at `http://nant.sourceforge.net`.

Chapter 2: Deployment Options

In This Chapter

- **Deploying Windows and Web applications using ClickOnce**
- **Digitally signing your code and strong naming assemblies**
- **Creating an installer**
- **Precompiling Web sites**

Deploying an application can be as simple as copying a single file or as complex as installing multiple components with dependencies on each other. Deployment can be complex at times, involving multiple directories and registry entries. Deploying needs to be done not just once, but throughout the life of your application. For example, you often need to deploy the following:

- Prerelease versions of your application for testing
- Major releases
- Updates to your application

Even though this deployment chapter comes near the end of the book, deploying is one of the first things you start thinking about for your application. You should test your deployment strategy with your daily build.

Understanding Deployment Basics

For too many clients, deployment consists of 25 steps that require you to do the following (not necessarily in this order):

- Copy file X from this server.
- Copy file Y from that server.
- Execute file X on the client's computer.
- Create six ODBC connections.
- Register component Z on the client's computer.
- Create shortcuts on the desktop and on the Start menu.

Requiring your customers or internal IT staff to jump through these hoops every time a client installation is required isn't only brutal — it's unprofessional. People will invariably make mistakes when they dread installing your application. Nothing is worse than discovering that you forgot to install a required component *after* the computer is deployed 20 miles away.

WARNING!

Complicated installation processes can quickly overwhelm even the smallest companies. Many IT shops use ghosting software that allows them to make an image of a client computer's hard drive and copy that image onto other machines. When a technician skips a step in manual deployment, the mistake multiplies exponentially when the ghosted image is used to build new machines.

TIP

The time to think about the ease or difficulty of installing software is before the software is ever built or acquired. If you're acquiring software, ask what is required to install it. If you're building commercial software, treat simplified deployment as a competitive advantage.

Some software development houses create an opportunity for others to make a lot of extra revenue by having sloppy deployment procedures, thus creating the need for consultants and third-party technical support. Using sloppy deployment procedures doesn't necessarily mean that you have to pay a consultant to help you configure new software, but it does mean that you have to jump through a bunch of hoops just to get the software installed.

Visual Studio provides two strategies for deploying your applications and components:

- **ClickOnce:** Publish your application or component to a common repository, where users can download the application.
- **Windows Installer:** Create an executable setup file that walks users through the steps in an installation wizard.

The strategy you choose depends on many factors, such as

- The complexity of your application
- The frequency of updates and releases
- Whether your application needs to run locally
- Whether users have network access
- Whether your application is all managed code

This chapter discusses the deployment options available in Visual Studio 2010, tells you when you should use each one, and then shares some third-party alternatives you should consider.

Deploying Smart Client Applications

In the late 1990s, the development world went crazy for Web applications. Part of the draw was capturing the ease of administration that comes along with Web sites. Rather than install a single Windows application on a thousand computers, why not have a thousand computers access a single Web application? It sounds great in theory; however, many folks quickly realized that using Windows applications sometimes just makes the most sense.

Microsoft listened and created a hybrid deployment model that allows you to deploy your Windows application to a Web server. Using the ClickOnce feature, Visual Studio 2010 copies the files needed to deploy your Windows application to a Web server. Your Windows clients then install, or even just run, the application from the Web server.

Of course, sometimes your application needs to exercise more control over the target installation machine than you can with a Web server. When you need more control over your application's installation, you may want to use *Windows Installer,* which is a component in Windows operating systems that manages software installation. Visual Studio 2010 provides several project templates you can use to target the services of Windows Installer.

From No Touch to ClickOnce

Microsoft started introducing the concept of Web-based installation in previous versions of the .NET Framework. The technology was improved, and the name changed from No Touch to ClickOnce.

You use ClickOnce to

- Copy application files to a central Web site.
- Deploy updates to a Web site.
- Install applications on client machines from the Web site.

You aren't restricted to deploying to a Web site; you can also deploy to an FTP server or to CD. You can configure an application to automatically check a Web site for updates. Alternatively, you can use the `System.Deployment` namespace to write custom code that targets ClickOnce.

You can choose for your clients to run an application locally or only from the Web server. Either way, the application is downloaded to a local cache. You can find the cache at `Documents and Settings\[`*`username`*`]\Local Settings\Apps`.

The user is supplied with a link to the Web server or file server. For example, a user can click a link that you send by e-mail so that the application is downloaded to a local cache and executed. The user doesn't need administration privileges because the application runs in a secure mode and can't access the machine.

A ClickOnce process using Visual Studio may work like this:

1. **The developer publishes a Windows application to a deployment server.**
2. **Visual Studio 2010 creates a Web site to act as a deployment server, and all files needed to run the application are copied to the Web site.**
3. **The client clicks a hyperlink that deploys the Windows application on the local machine.**

   The hyperlink is a link to a setup file on the deployment server. The setup file downloads the application's files to the local computer and executes the application.

You have two options for deploying the application on the client:

- **Online:** The user must click the hyperlink every time she wants to run the application. The application is always installed in the local cache. When the user clicks the link, the deployment server downloads any updated components before executing the application from the local cache.
- **Offline:** The user can choose a shortcut from the Start menu to open the application. An uninstaller is provided via Control Panel.

The good news is that you don't have to do anything special to use ClickOnce. The features are part of the .NET Framework. The bad news is that you can't install all applications by using ClickOnce. For example, you can't access the Registry or add assemblies to the Global Assembly Cache (GAC) by using ClickOnce. Use Windows Installer instead.

ClickOnce is intended for scenarios where your applications use a rich user interface (UI) to provide access to resources, such as database servers and Web services. Many corporate applications fit into this scenario, which makes ClickOnce helpful from an administration standpoint. You don't have to physically deploy or update applications on every machine.

Visual Studio 2010 provides the Publishing Wizard for taking advantage of the .NET Framework ClickOnce technology. You have two options for accessing the Publishing Wizard:

- Choose Build➪Publish.
- Use the Publish tab in Project Designer.

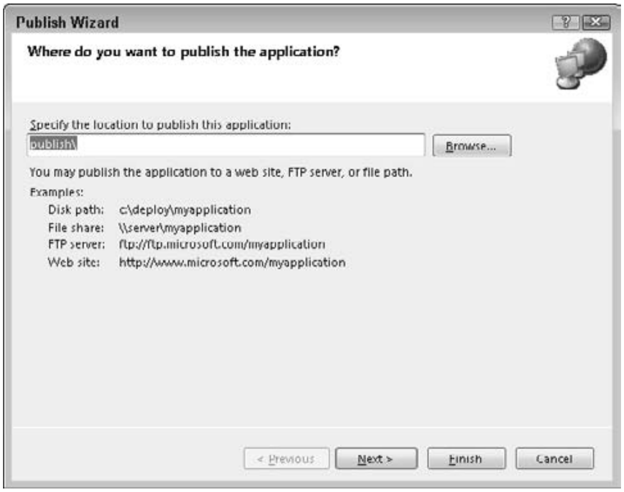
Applications published via the Publishing Wizard are *ClickOnce applications.*

To publish a Windows application to a Web site by using the Publishing Wizard, follow these steps:

1. **Open the Windows application in Visual Studio.**
2. **Choose Build➪Publish <*projectname*>.**

   Your project name appears after the Publish command on the Build menu. The Publish Wizard appears.

   The default choice is to publish to a publish subfolder of the solution folder (see Figure 2-1), and the wizard lists a location to create the Web site. If you want to use a different URL, type the pathname in the wizard.



Publish Wizard

Where do you want to publish the application?

Specify the location to publish this application:

publish\

You may publish the application to a web site, FTP server, or file path.

Examples:

Disk path: c:\deploy\myapplication

File share: \\server\myapplication

FTP server: ftp://ftp.microsoft.com/myapplication

Web site: http://www.microsoft.com/myapplication

Figure 2-1: You can specify a URL for the publish destination.

   Alternatively, you can type or browse to a file share, FTP server, or disk location.

3. **Click Next.**
4. **At the prompt, choose whether users will install the application from a web site, UNC path or file share, or CD/DVD-ROM media.**
5. **Click Next.**

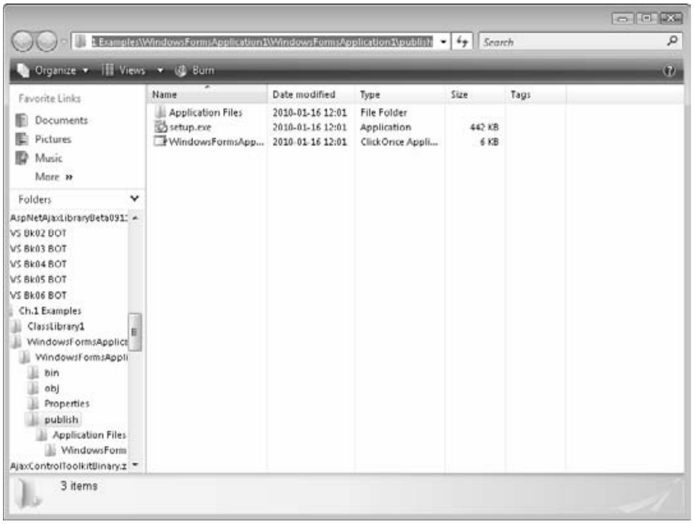
   The Where Will the Application Check for Updates page appears. By default, the application won't check for updates. You can also specify a location where the application will check for updates.

6. **Click Next**

   The Ready to Publish page appears with a summary of your choices.

7. **Verify your choices and click Finish.**

   The wizard builds your project and publishes it to the Web server. If your application is published to a file, a folder directory will open, showing you the target, as shown in Figure 2-2.



1 Examples\WindowsFormsApplication1\WindowsFormsApplication1\publish

Name	Date modified	Type	Size	Tags
Application Files	2010-01-16 12:01	File Folder		
setup.exe	2010-01-16 12:01	Application	442 KB	
WindowsFormsApp...	2010-01-16 12:01	ClickOnce Appli...	6 KB	

Figure 2-2: Here you can see the target that has been saved to disk.

Accessing the published application

When you publish the application to a Web server, ClickOnce creates a Web site with a `publish.htm` file you can use to deploy the application to the local client. Accessing a published application is often referred to as *consuming the application.*

To access the published application locally, follow these steps:

1. **Browse to the URL where the application was published.**

   You can find the URL in the output window, as described in the preceding section. You can also view the Publish tab in Project Designer to determine where the application is published.

   If you publish your application to a file share, use the UNC path to access the `publish.htm` file. If you publish to a CD, run the `setup.exe` file.

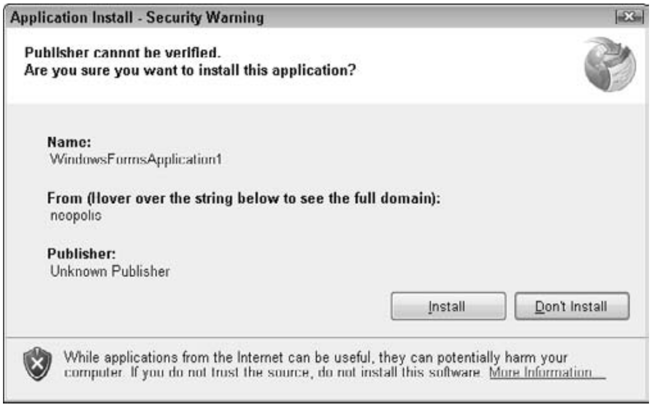
2. **Click the Install button on `publish.htm` or `setup.exe`.**

   On machines that use Vista as the operating system, you either need the install application to be signed, or you need to use administrative privileges.

ClickOnce downloads local copies of the application to a cache and launches the application. A security warning appears if the application isn't properly signed, as shown in Figure 2-3. See the later section "Signing your code" for more information on removing this security warning.

3. **Click the Install button to complete the installation.**

   The files are copied, and the application is launched.



Application Install - Security Warning

Publisher cannot be verified.
Are you sure you want to install this application?

Name:
WindowsFormsApplication1

From (Hover over the string below to see the full domain):
neopolis

Publisher:
Unknown Publisher

While applications from the Internet can be useful, they can potentially harm your computer. If you do not trust the source, do not install this software. More Information

Figure 2-3: A security warning appears for unsigned applications.

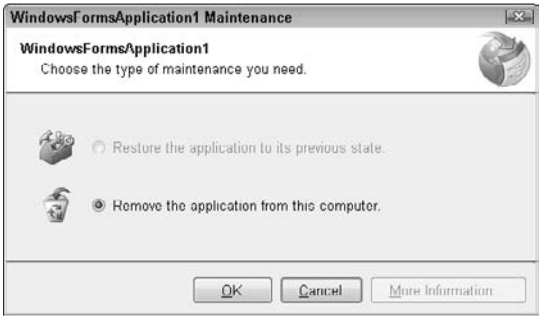
If you chose to publish your application so that it runs offline, the installation process creates (on the Start menu) a shortcut to the cached copy of the application. Otherwise, you can run the application only by using the URL to the deployment server. You can remove or configure the application by using the Add/Remove Programs applet in Control Panel.

To remove an installed application by using Control Panel, follow these steps:

1. **Open Control Panel.**
2. **Double-click the Add or Remove Programs icon.**
3. **Navigate to the program you want to remove in the Add or Remove Programs window.**
4. **Click the Change/Remove button.**

   A maintenance window appears, as shown in Figure 2-4.
5. **Select the radio button to remove the application and then click OK.**

   The application is removed from the local computer.



WindowsFormsApplication1 Maintenance

WindowsFormsApplication1

Choose the type of maintenance you need.

○ Restore the application to its previous state.

● Remove the application from this computer.

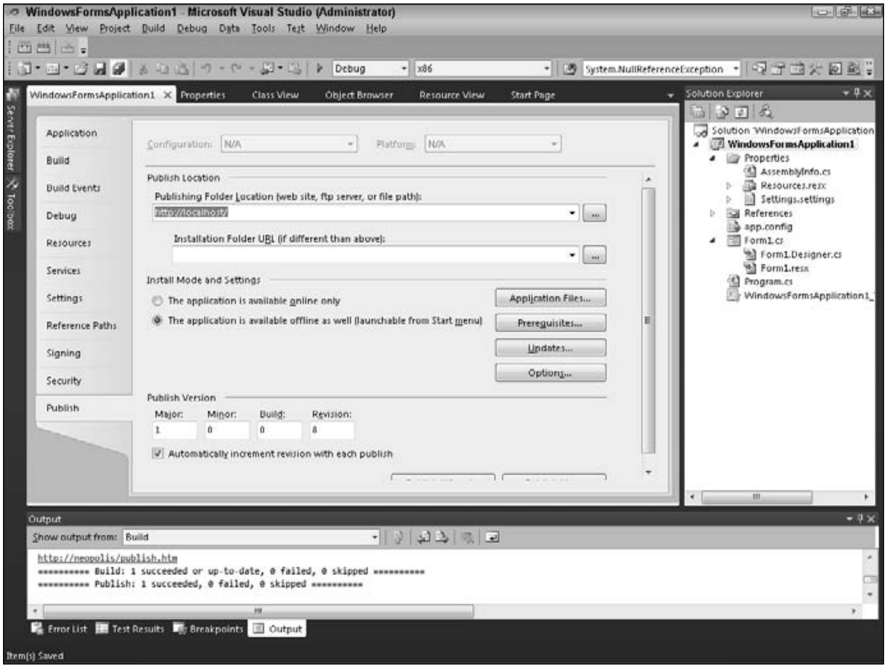
Figure 2-4: You can uninstall the application here.

Checking out your publishing options

Visual Studio provides several options for deploying applications with ClickOnce. You access these options from the Publish tab in Project Designer. To access the Publish tab, follow these steps:

1. **Right-click the project in Solution Explorer.**
2. **Choose Properties from the shortcut menu.**

   Project Designer appears.
3. **Click the Publish tab, shown in Figure 2-5.**



WindowsFormsApplication1 - Microsoft Visual Studio (Administrator)

Configuration: N/A Platform: N/A

Publish Location

Publishing Folder Location (web site, ftp server, or file path): http://localhost/

Installation Folder URL (if different than above):

Install Mode and Settings

○ The application is available online only

● The application is available offline as well (launchable from Start menu)

Publish Version

Major: 1 Minor: 0 Build: 0 Revision: 8

☑ Automatically increment revision with each publish

Show output from: Build

```
http://neopolis/publish.htm
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
========== Publish: 1 succeeded, 0 failed, 0 skipped ==========
```

Item(s) Saved

Figure 2-5: Set the publish specifications here.

You can use the Publish tab to do the following:

- **Set the publishing location.** Specify the path to a Web server, FTP server, network file share, or local file path where you want to publish files.
- **Designate the installation as online or offline.**
- **Specify which application files to include in the deployment.**
- **Select prerequisite components to install on the client machine.** By default, version 4 of the .NET Framework is required. You can choose to require your own custom components or components from third parties, such as Crystal Reports.
- **Schedule application updates and provide an alternative location for retrieving updated files.**
- **Set options, such as a URL for support, or make an application deployed to a CD automatically start when the CD is inserted.**
- **Specify that the Publishing Wizard automatically increments version numbers each time the application publishes.**

Signing your code

For a ClickOnce application to run on a local computer, the computer needs to know that the application is safe to run. ClickOnce accomplishes this by digitally signing the application, also known as *code signing.* Digitally signed applications bind an application's publisher to the application and ensure that the application hasn't been tampered with en route to the consumer.

A digitally signed application uses a digital certificate to store all the information required to validate an application and its publisher. When you use the Publishing Wizard, Visual Studio automatically creates a certificate you can use for testing. Common practice dictates using a separate certificate for testing and releasing software to prevent someone from obtaining test code with a release certificate.

Code signing is a fairly common requirement. You should create a single test certificate and reuse it rather than allow Visual Studio to create a new certificate every time. To sign a ClickOnce application, follow these steps:

1. **Right-click the project in Solution Explorer.**
2. **Choose Properties from the shortcut menu.**

   Project Designer appears.
3. **Click the Signing tab.**

   The Signing tab has two purposes:

- Sign ClickOnce files.
- Sign assembly files.

See the upcoming section "Sharing assemblies" for more information on signing assembly files.

4. **Select the Sign the ClickOnce Manifests check box.**

   If you've already run the ClickOnce Wizard, Visual Studio automatically creates a test certificate. You see a check mark and the certificate's details, as shown in Figure 2-6.

5. **Click Select from Store or Select from File to select a certificate to use.**

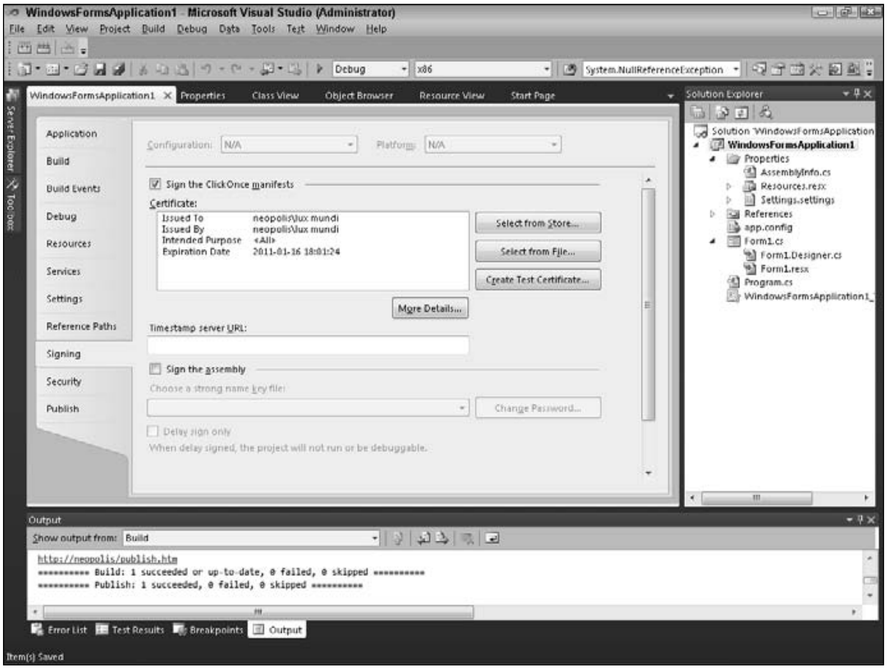
   If you need to create a new test certificate

   a. *Click the Create Test Certificate button.*

      You're prompted for a password to use for the certificate. Visual Studio creates the certificate and adds it to the project. The certificate is installed in your computer's personal store of certificates.

   b. *Copy the `.pfx` file (personal exchange file) to a central location so that you can reuse it.*

      Next time, you can choose Select from File and select the test certificate.



Configuration: N/A Platform: N/A

☑ Sign the ClickOnce manifests

Certificate:

Issued To	neopolis\lux mundi
Issued By	neopolis\lux mundi
Intended Purpose	<All>
Expiration Date	2011-01-16 18:01:24

Timestamp server URL:

□ Sign the assembly

Choose a strong name key file:

□ Delay sign only

When delay signed, the project will not run or be debuggable.

Show output from: Build

```
http://neopolis/publish.htm
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
========== Publish: 1 succeeded, 0 failed, 0 skipped ==========
```

Figure 2-6: You can digitally sign your application.

Alternatively, you can use the `makecert.exe` command in the .NET Software Development Kit to create certificates.

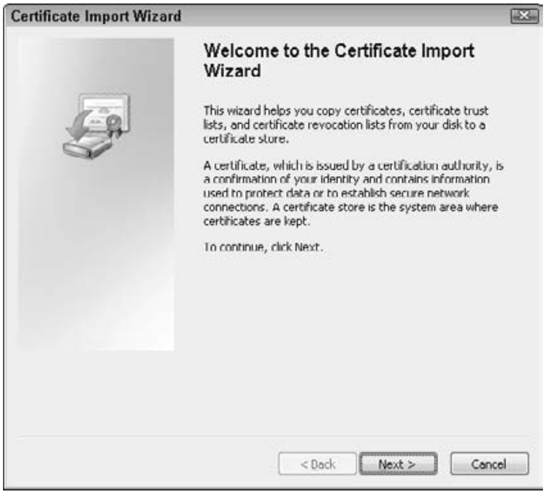
6. **Click the More Details button.**

   The Certificate window appears.

   The text in the certificate states that the certificate isn't trusted. At this point, a certificate in your personal store doesn't come from a trusted certificate authority (CA). You must install the certificate in the Trusted Root Certificate Authorities store.

7. **Click the Install Certificate button, as shown in Figure 2-7.**

   The Certificate Import Wizard appears.



Certificate Import Wizard

Welcome to the Certificate Import Wizard

This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

Figure 2-7: You must invoke the Certificate Import Wizard.

8. **Click Next to step through the wizard.**

   The Certificate Store page appears.

9. **Select the Place All Certificates in the Following Store option.**

10. **Click the Browse button.**

    The Select Certificate Store window appears.

11. **Select Trusted Root Certificate Authorities and then click OK.**

    The store appears in the wizard.

12. **Click Next.**

13. **On the Completion page, click Finish.**

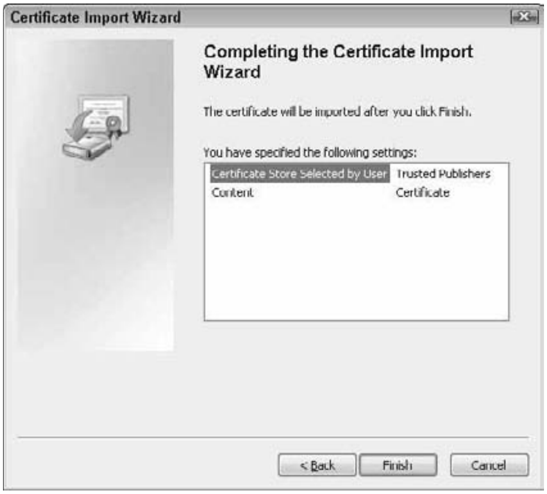
A security warning informs you that you're about to install a new trusted certificate authority. The danger is that someone can sign software with your test certificate and run the software on your computer.

14. **Click Yes to accept the security warning.**

    The certificate is installed in the Trusted Root Certificate Authorities store.

15. **Repeat Steps 7 through 13 to install the certificate in the Trusted Publishers store and click Finish.**

    Figure 2-8 shows a certificate published in the Trusted Publishers store.



Certificate Import Wizard

Completing the Certificate Import Wizard

The certificate will be imported after you click Finish.

You have specified the following settings:

Certificate Store Selected by User	Trusted Publishers
Content	Certificate

Figure 2-8: This shows the certificate in the Trusted Publisher store.

Now, whenever you install the application, you aren't prompted to install it.

WARNING!

You must use Internet Explorer to sign up for its certificate. At this time, Firefox and other browsers don't appear to work.

Obviously, you don't want to deploy your test certificate into production. You need to acquire a release certificate from a CA, which is an entity whose identity has been verified and who can validate your identity. For example, VeriSign is a CA from whom you can buy a certificate. Read more about the VeriSign code-signing products on its Web site:

```
www.verisign.com/products-services/security-services/code-signing
```

If you're distributing software only within your internal organization, you may not want to purchase a certification from a third party. In that case, your organization can make itself a CA for within your organization and issue certifications.

Whichever route you take, you need some way to distribute the certificates to the Trusted Publishers store on all your clients. Additionally, if your company is issuing its own certificates, it needs to register itself on each client machine in the Trusted Root Certification Authorities store.

TIP

You probably don't want to run around to each client machine and install code-signing certificates. That defeats the purpose of using ClickOnce. Instead, use Windows Installer to create an installer package that installs the certificates. You can use other tools, too, such as the Windows certificate-management console (`certmgr.exe`).

If you want to read more about code signing, check out the Microsoft white paper on code-signing best practices at

`www.microsoft.com/whdc/winlogo/drvsign/best_practices.mspx`

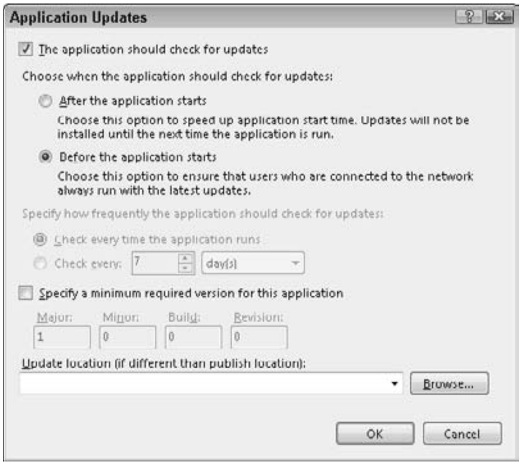
Updating your applications

You have several options for updating your ClickOnce applications. When you need to publish updates, you just update your software in Visual Studio and publish it again via the Publishing Wizard. Old versions of the software are retained, and new versions are pushed out.

When and how frequently a user's application checks for updates depends on how you configure the Publishing Wizard. You access update options by using the Updates button on the Publish tab of Project Designer. See the earlier section "Checking out your publishing options" for more information on using the Publish tab.

These update choices are available to you:

- **After the application starts:** Speeds up start time because the application checks for updates after it opens. Any new updates are downloaded and installed the next time the application runs.
- **Before the application starts:** Updates are installed when a user launches the application.
- **Scheduled updates:** Indicates how frequently the application should check for updates, as shown in Figure 2-9.



Application Updates

☑ The application should check for updates

Choose when the application should check for updates:

○ After the application starts

Choose this option to speed up application start time. Updates will not be installed until the next time the application is run.

● Before the application starts

Choose this option to ensure that users who are connected to the network always run with the latest updates.

Specify how frequently the application should check for updates:

● Check every time the application runs

○ Check every: 7 day(s)

□ Specify a minimum required version for this application

Major: 1 Minor: 0 Build: 0 Revision: 0

Update location (if different than publish location):

Figure 2-9: Here, updates are scheduled every seven days.

- ✦ **Required update to a minimum version:** Forces a client to update by requiring the use of at least a minimum version of the application.

You can use the Application Updates window to specify an update location that's different from a publish location. For example, you can publish to a CD and update to a Web server.

Each time you run the Publishing Wizard, Visual Studio automatically updates the application's version number. The version number is part of the application's digital signature, which .NET uses to ensure that the application is valid. If you haven't made any changes to the application, the build process doesn't build the application. As a result, the wizard uses a new version number when it publishes a copy of the application from the previous version. As a result, the application doesn't run on the client machine. If you want to force Visual Studio to publish an update when you haven't made any changes to the code, be sure to use the Build➪Clean command first. The Publishing Wizard is then forced to build the application.

Making a Windows Installer

Visual Studio 2010 provides setup projects you can use to deploy your applications. Setup projects create a Windows Installer file with the extension `.msi`.

These setup projects are available in Visual Studio:

- ✦ **Setup projects create an installer file** that you can use to deploy a program on a target machine or to a virtual directory on a Web site. Visual Studio provides a wizard to walk you through creating setup projects.

- **Merge module projects create reusable setup components** for consumption by setup projects.
- **A Cab project creates a CAB file** from the items added to Solution Explorer. *CAB files* are often used to distribute components, such as ActiveX controls. The `.cab` file format (short for Microsoft Cabinet) is used for compressing files.

You can distribute the resulting MSI (Microsoft Installer) files by using CDs or DVDs, or a file share for installation across a network. MSI files can install an application on a target machine in a Windows application or on a Web site in a Web application.

The installation projects provided by Visual Studio take advantage of Windows Installer, which is an installation framework available on Windows operating systems. The installer takes care of how everything gets installed; you just focus on what you need to install. For example, the installer can create Registry entries, shortcuts, and Open Database Connectivity (ODBC) connections for you. You use a Visual Studio setup project to tell Windows Installer what to install. You don't have to write the code to make any of this installation happen. Windows Installer gets it installed for you.

Applications installed by using Windows Installer can

- Self-repair if the program is damaged.
- Roll back if the installation can't be completed.

Windows Installer is sophisticated, and there's nothing you can't install with it.

Creating a setup project for an existing application

Although you can create standalone setup projects, you usually add setup projects to an existing solution that you want to deploy. To create a setup project for an existing application, follow these steps:

1. **Open a solution that includes a project you want to deploy.**
2. **Right-click the solution in Solution Explorer.**

   A shortcut menu appears.
3. **Choose Add⇨New Project.**
4. **In the Add New Project window, expand the Other Project Types category and select Setup and Deployment.**

   You can choose either Install Shield 2010 or Windows Installer. If you choose Windows Installer, a list of project templates appear.

5. **Select Setup Wizard.**
6. **Type a name for the project and then click OK.**

   The Setup Wizard appears.
7. **Click Next.**

   The Choose a Project Type page appears.
8. **Select the Create a Setup for a Windows Application option.**

   You can use the wizard to create a Web setup project, a merge module, or a CAB file.
9. **Click Next.**

   The Choose Project Outputs page appears. The project output from the other projects in the solution appears on this page. If no other projects are in the solution, this page doesn't appear. You have to add files to the setup project manually.
10. **Select the project output groups you want to deploy with your setup project.**

    You want, at a minimum, content files and the primary output, which is the assembly created by your project.

    TIP

    Specify that a file should be included as a content file by accessing the file's properties in Solution Explorer.
11. **Click Next.**
12. **Add any files you want to deploy with your setup project.**
13. **Click Next.**

    The Create Project page appears.
14. **Click Finish.**

    The project is created, and the content is added to Solution Explorer, as shown in Figure 2-10.

Setup projects have a number of editors you can use to build your installer, including the ones in this list:

- **File System:** Identifies which files and shortcuts are created on the file system of the target machine
- **Registry:** Creates Registry entries
- **File Types:** Associates file extensions with applications
- **User Interface:** Customizes the installation wizard
- **Custom Actions:** Customizes the installation process
- **Launch Conditions:** Creates conditional installation steps

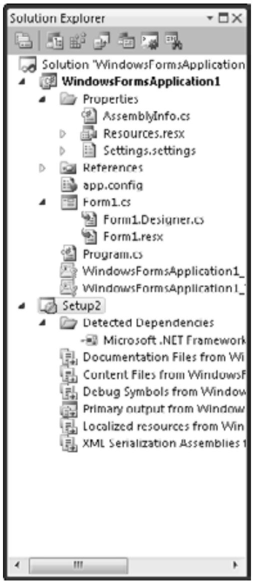

Solution Explorer
Solution 'WindowsFormsApplication
WindowsFormsApplication1
Properties
AssemblyInfo.cs
Resources.resx
Settings.settings
References
app.config
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
WindowsFormsApplication1_
WindowsFormsApplication1_
Setup2
Detected Dependencies
Microsoft .NET Framework
Documentation Files from Wi
Content Files from WindowsF
Debug Symbols from Window
Primary output from Window
Localized resources from Win
XML Serialization Assemblies f

Figure 2-10: Setup Wizard adds the new setup project in Solution Explorer.

These editors are fairly straightforward to use after you realize that you're taking the perspective of the target computer. You have to start by asking yourself what you want the target machine to look like when the installer is complete and then figure out how to make that happen by using these editors.

Putting the installer to work

To install a simple application and create shortcuts on the user's desktop and Start menu, use the File System Editor:

1. **Create a new setup project, as described in the preceding set of steps.**

   The File System Editor opens by default.

   You can open any of the editors by using the View menu or the icons on the Solution Explorer toolbar.

   The File System Editor displays the file system of the target machine. You can elect to install files or shortcuts to special folders that are common to all Windows computers, such as a User's Desktop or Favorites folder.

   By default, the Application Folder, User's Desktop, and User's Program Menu folders are added to the editor. Right-click the File System on Target Machine option to add more folders.

2. **Select Application Folder.**

   The folder's contents appear in the pane on the right.

If you added your setup project to a solution with an existing project, the projects' output selected in the wizard appears. Right-click Application Folder and choose from the menu to add more files.

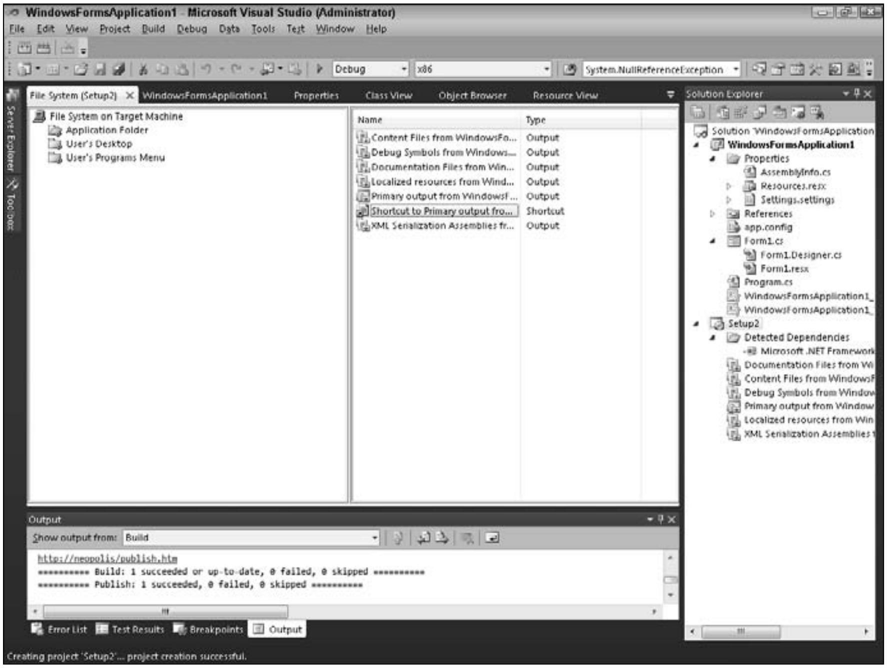
Use the Properties window to set an installation folder for Application Folder.

3. **Right-click Primary Output in the Application folder.**

   A shortcut menu appears.

4. **Choose Create Shortcut.**

   A new shortcut appears, as shown in Figure 2-11.



Name	Type
Content Files from WindowsFo...	Output
Debug Symbols from Windows...	Output
Documentation Files from Win...	Output
Localized resources from Wind...	Output
Primary output from WindowsF...	Output
Shortcut to Primary output fro...	Shortcut
XML Serialization Assemblies fr...	Output

```
http://neopolis/publish.htm
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
========== Publish: 1 succeeded, 0 failed, 0 skipped ==========
```

Creating project 'Setup2'... project creation successful.

Figure 2-11: Adding shortcuts is easy, and they show in this list.

5. **Type a name for the shortcut.**

   You want the shortcut to appear on the target computer.

6. **Right-click the shortcut and choose Cut.**

7. **Right-click User's Desktop and choose Paste.**

   The shortcut appears in the folder.

8. **Right-click the User’s Programs Menu folder.**
9. **Choose Add➪Folder from the shortcut menu.**
10. **Type a name for the folder.**

    You want the folder to appear on the user’s Program menu.
11. **Repeat Steps 3 through 7 to add a shortcut to the new folder.**

You now have an installer that installs your application and creates shortcuts on the user’s desktop and then adds a folder with a shortcut to the Program menu.

Here’s how to test the installer:

1. **Right-click the setup project in Solution Explorer.**
2. **Choose Build from the shortcut menu.**

   The project builds and packages the files.

   If you choose to build the entire solution, the setup project isn’t included in the solution’s build configuration in Configuration Manager. Either enable it to be built or build the project from the Build menu.
3. **Repeat Step 1 and choose Install from the shortcut menu.**

   The installation wizard appears.
4. **Step through the wizard to install your application.**

Sharing assemblies

Most of the time, you want your assemblies to be private. Only your application can access private assemblies. If you want to share your assemblies with other applications, you need to place them in the GAC. Examples of assemblies you may find in the GAC are the assemblies of the .NET Framework. When you place the assemblies in the GAC, all applications have access to them.

To publish an assembly to the GAC, it must use a strong name, which guarantees that all assemblies in the GAC have a unique name. A *strong name* comprises the assembly name, version number, public key, and digital signature. Suppose that your company creates an assembly named `OrderEntry`. Another company could feasibly create an assembly with the same name. A strongly named assembly creates a unique assembly, even if the assembly names were identical.

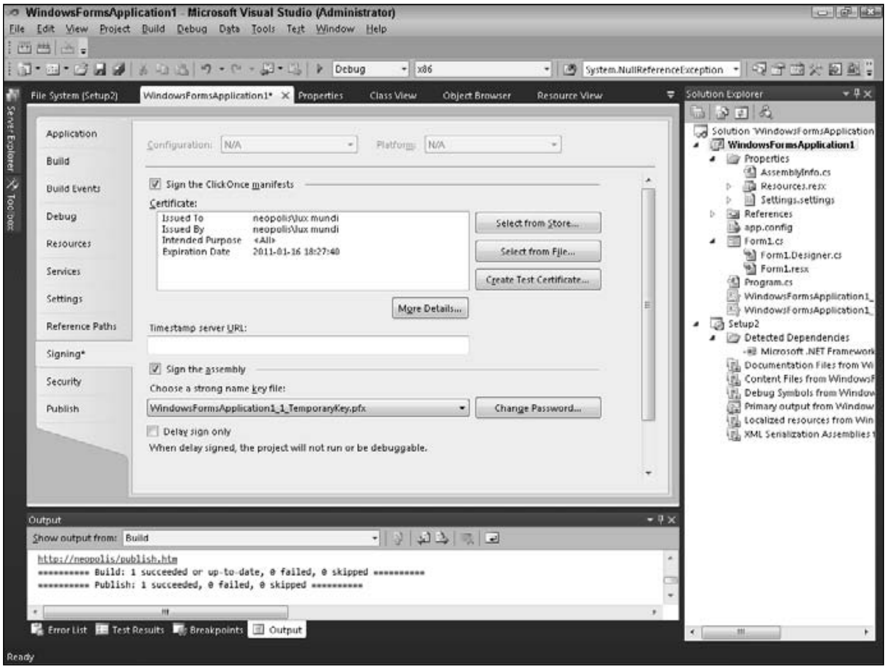
If you attempt to publish to the GAC an assembly that isn’t strongly named, you receive an error message.

Visual Studio 2010 provides a way to strongly name assemblies and publish them to the GAC. To strongly name an assembly, follow these steps:

1. **Go to the Signing tab of Project Designer for the assembly you want to sign.**
2. **Select the Sign the Assembly option.**
3. **Open the Choose a Strong Name Key File drop-down list.**
4. **Click Browse to reuse an existing key file or click New to create a new key file.**

   If you click New, the Create Strong Name Key window appears. Type a name for the key file and a password to protect the file.

   The key file appears on the tab, as shown in Figure 2-12.



Configuration: N/A   Platform: N/A

☑ Sign the ClickOnce manifests

Certificate:

```
Issued To          neopolis\lux mundi
Issued By          neopolis\lux mundi
Intended Purpose   <All>
Expiration Date    2011-01-16 18:27:40
```

Timestamp server URL:

☑ Sign the assembly

Choose a strong name key file: WindowsFormsApplication1_1_TemporaryKey.pfx

☐ Delay sign only

When delay signed, the project will not run or be debuggable.

Show output from: Build

```
http://neopolis/publish.htm
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
========== Publish: 1 succeeded, 0 failed, 0 skipped ==========
```

Figure 2-12: The strong key file has been added.

To install the assembly in the GAC by using a setup project, follow these steps:

1. **Open the File System Editor in the setup project.**

2. **Right-click File System on Target Machine.**

   A shortcut menu appears.

3. **Choose Add Special Folder⇨Global Assembly Cache Folder.**

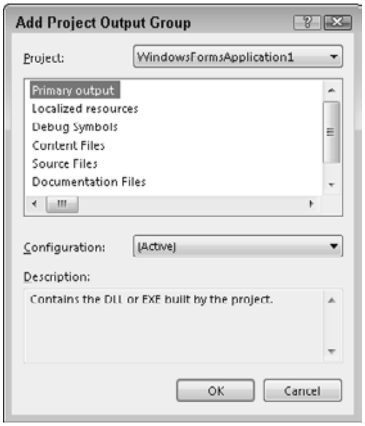
   A GAC folder appears in the editor.

4. **Right-click Global Assembly Cache Folder in the editor.**

5. **Choose Add⇨Project Output.**

   The Project Output Group window appears.

6. **Select the project from the Project drop-down list and then choose Primary Output, as shown in Figure 2-13.**



Project: WindowsFormsApplication1

Configuration: (Active)

Description: Contains the DLL or EXE built by the project.

Figure 2-13: The project will install to the GAC.

7. **Click OK.**

   The primary output appears in the editor.

Build the setup project and install it on a test machine. To view the GAC to ensure that the assembly was installed, take these steps:

1. **Open Administrative Tools in Control Panel.**

2. **Double-click Microsoft .NET 4 Configuration.**

   The configuration management console appears.

3. **Select Manage the Assembly Cache.**

4. **Select the View List of Assemblies in the Assembly Cache option.**

   A list of assemblies appears.

Deploying Web Applications

Deploying Web applications is more complicated than deploying Windows applications. One key difference is what occurs during the build process. Windows applications are compiled into assemblies. You can distribute those assemblies by any means you want. Visual Studio provides a number of tools to assist with the distribution of Windows assemblies, as described in the preceding section.

Web sites are different. When you use the Build command with a Web site, Visual Studio compiles the page only for testing. You can't deploy the output created from the Build command — and you don't need to. Because ASP.NET Web sites are compiled on demand when users access the Web site, you can just copy and paste your Web site, if you want.

You may want to compile your Web sites before you deploy them, which is often called *precompiling.* Precompiled Web sites have these benefits:

- They start faster because no performance hit occurs while the requested page compiles.
- Source code isn't deployed to the Web server; only assemblies and static files are.
- You get feedback from compiler errors before the Web site is deployed.

Next, the deployment begins. Web sites often have two stages of deployment:

- **Test server:** When a Web site is first created, it may be initially deployed to a test server. Several developers and testers can access the test site. The test site usually remains up after the site is deployed to production. The test site provides a working copy of the production Web site, where changes can be made.
- **Production server:** After a site is ready to go live, it's published to a production Web server.

In reality, several versions of the Web site can run on different test servers. Developers can also have local copies of the site running on their development machines. They may deploy their local copies to the test server.

Because of the nature of Web sites, they're often changed more frequently than Windows applications. Changes are usually made to a single page or a set of pages, and then only those changes are deployed. In other cases, the entire Web site may be deployed.

Making frequent changes to the structure, layout, and design of a Web site can be disruptive to your site's visitors. As a result, more firms are starting to limit the number of changes they make to the site. Rather than change the site, firms are creating fluid sites that use content databases and configuration files to control the site's content and flow. Rather than change the Web site when you want to change content or the site flow, changes are made to content databases or configuration files.

Whether you're deploying to a test server or a production Web server, Visual Studio provides several options for deploying Web sites:

- **Web setup project:** Packages your Web site by using an installer that you then execute on the Web server
- **Copy Web Site tool:** Copies and synchronizes sites between a local store and a server
- **Publish Web Site:** Precompiles the Web site and copies it to the target of your choice
- **Web Deployment Projects:** This add-in gives you the most flexibility in precompiling your Web site

You can also just copy and paste your files from your test server to production. The major drawback of copy-and-paste strategies is that you can easily make mistakes.

Using the Publish Web Site utility

To use the Publish Web Site utility in Visual Studio, follow these steps:

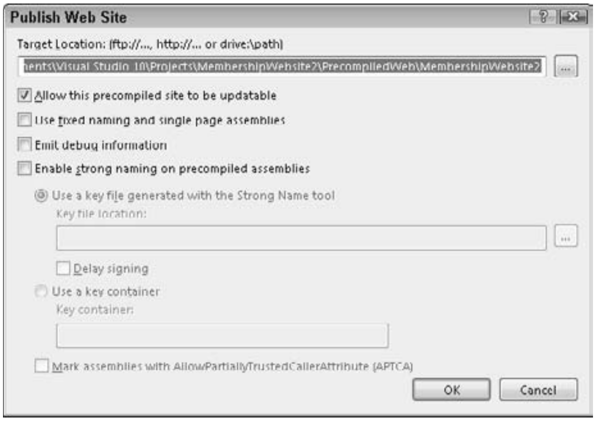
1. **Open your Web site in Visual Studio.**
2. **Choose Build⇨Publish Web Site.**

   The Publish Web Site dialog box, shown in Figure 2-14, appears.
3. **Specify a target location where the Web site should be copied.**

   You can enter or browse to a URL, Universal Naming Convention (UNC), or local file path.
4. **Select the Allow This Precompiled Site to Be Updateable option.**

   This feature copies your ASPX files as-is to the Web site. You can update these pages later without having to recompile and deploy the entire site.
5. **Click OK.**

   All code-behind files are compiled into a single assembly. The assembly, ASPX files, and static files (such as images) are copied to the target location that you specify in Step 3.



Publish Web Site

Target Location: (ftp://..., http://... or drive:\path)

hents\Visual Studio 10\Projects\MembershipWebsite2\PrecompiledWeb\MembershipWebsite2

☑ Allow this precompiled site to be updatable

□ Use fixed naming and single page assemblies

□ Emit debug information

□ Enable strong naming on precompiled assemblies

● Use a key file generated with the Strong Name tool

Key file location:

□ Delay signing

○ Use a key container

Key container:

□ Mark assemblies with AllowPartiallyTrustedCallerAttribute (APTCA)

Figure 2-14: All necessary files will be published with your Web site.

Make a backup of your production Web site before deploying a new Web site. If the new site doesn't work, you can quickly roll back.

Handling web.config files for testing and production

A common challenge when you're deploying Web sites is not overwriting the `web.config` file. The `web.config` file on a test server usually has different settings than on a production server. For example, the test server may have connection strings that point to test databases. Visual Studio 2010 automatically creates a `web.debug.config` and `web.release.config` file for you and gives you control over precompilation and other deployment settings.

To configure deployment of an ASP.NET Web project, follow these steps:

1. **Choose Build➪Publish Web Site.**

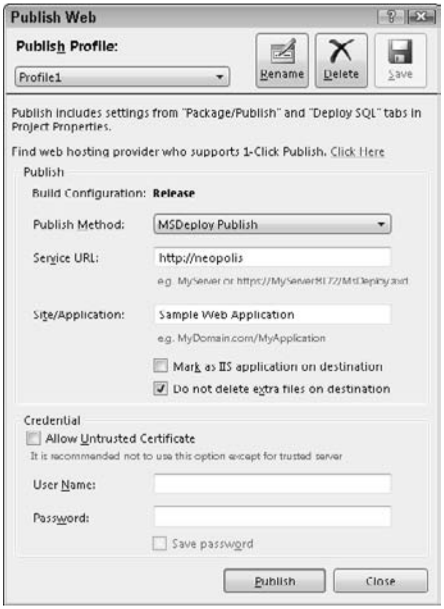
   The Publish Web dialog box appears, as shown in Figure 2-15.

2. **Choose Publish.**

   Visual Studio creates a `web.release.config` file and publishes the Web site with your settings.

You configure a group of property page settings for each build configuration. For example, you likely don't want to use the deployment project with your debug build configuration. You can create different deployment settings for staging and release build configurations.

Build your site by using the release configuration. The `web.config` file is merged with the `.config` file. And you have a release version of your Web site that you can deploy to your production server, using whatever method suits you.



Publish Profile: Profile1

Publish includes settings from "Package/Publish" and "Deploy SQL" tabs in Project Properties.

Find web hosting provider who supports 1-Click Publish. Click Here

Publish

Build Configuration: **Release**

Publish Method: MSDeploy Publish

Service URL: http://neopolis

e.g. MyServer or https://MyServer8172/MsDeploy.axd

Site/Application: Sample Web Application

e.g. MyDomain.com/MyApplication

☐ Mark as IIS application on destination

☑ Do not delete extra files on destination

Credential

☐ Allow Untrusted Certificate

It is recommended not to use this option except for trusted server

User Name:

Password:

☐ Save password

Figure 2-15: The Publish Web dialog box.

Chapter 3: Checking Out Source Control

In This Chapter

- **Setting up source control**
- **Adding and retrieving source code**
- **Performing daily integration tasks**
- **Using source control on a team project**

Whether you're on a team of hundreds or a team of one, you need some way to manage your source code. You use source code control software to

- ✦ Provide a central repository for storing source code.
- ✦ Create a new version of source code every time changes are made.
- ✦ Track the history of your source code.

The standard and Professional versions of Visual Studio don't provide source code control features. However, they provide menus and windows that you can use to access some third-party, source-code control products, such as Perforce by Perforce Software or Microsoft's Visual Source Safe. Any source code control software that implements the Visual Studio Integration Protocol can provide source control services to Visual Studio. In this chapter, you explore using Perforce source code control with Visual Studio.

You can install Perforce source code control from `www.perforce.com`. You can use Perforce free of charge for up to two users and five client connections. Perforce installation and configuration is beyond the scope of this text.

Microsoft Visual Source Safe is a deprecated product from Microsoft and is no longer available for download. You can use existing installations of Visual Source Safe with Visual Studio 2010.

Using Source Code Control with Visual Studio

Visual Studio provides menu commands that you can use to drive the features of your integrated source code control provider. Visual Studio acts as a gateway to third-party source control providers, which are referred to as *plug-ins* in Visual Studio.

A source code control environment includes these elements:

- **A centralized database:** Stores master copies of source code
- **Local working folders:** Holds source code that's downloaded on each developer's computer for editing

A team can use a dedicated build server that downloads source code from the database to a local working folder.

How source code control works

Any developer who needs to use source code downloads a working copy of the source code to a local folder. To edit the source code files, the developer must check out the source code files. After the editing is finished, the developer checks the files back into the source code control database.

Checking a file into source code is often referred to as *integration.* Presumably, the developer has unit-tested the code before integrating it into the master source code. Unit testing your code before checking it into source code helps maintain the integrity of the master source code. Integrated code, even when it's tested, can cause the master source code not to compile. See Book III, Chapter 8 for more information on unit testing.

Automated build processes usually download the latest version of source code and execute a build script to build the master source code. The build server may create in the database a new version of the code, labeled with the build number. The development team can then re-create the build at any time. See Chapter 1 of Book VI for more information on build automation.

Getting ready for source control

Before you can integrate source code control with Visual Studio, you must do a couple of things first:

- **Set up and configure your source code control provider's server environment.** Setting up the server usually involves installing administration tools, adding users, and creating new databases.

- **Install your source code control provider on each client who needs access to the source control database.** Even though you access source code control via menu commands in Visual Studio, your source control provider's client software must be installed on each client computer.

After you install your source code control client, Visual Studio displays additional menus and commands that you can use to access your source code control repository. Here are some common source code controls tasks you can complete:

- Add a new or existing project to source control.
- Open an existing project from source control.
- Check out or check in source code.
- Get the latest version of source code.
- Remove a project from source control.

One of the first tasks you need to do before you start using source code control is setting your local working folder. The local working folder is the local repository on the client's machine where source code files are downloaded. When you edit source code files, you edit them by using files in the local working folder.

Your source-code control provider allows you to set a local working folder. If you don't set a local working folder, Visual Studio downloads files to your default projects folder. You can change the default projects folder in Visual Studio or set the local working folder in your source control provider.

To change the default projects folder in Visual Studio, follow these steps:

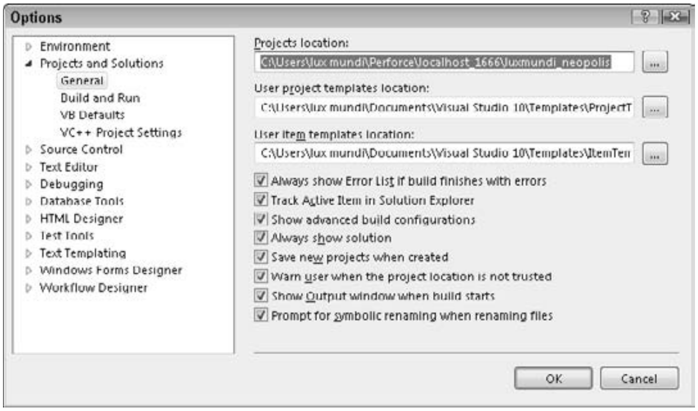
1. **Choose Tools➪Options.**

   The Options window appears.

2. **Select Projects and Solutions.**

   A list of settings for projects and solutions appears. Here, you provide the working directory that you used when you setup your workspace in Perforce as part of the installation.

3. **Set the default folder location by using the Visual Studio Projects Location field, as shown in Figure 3-1.**
4. **Click OK.**



Projects location: C:\Users\lux mundi\Perforce\localhost_1666\luxmundi_neopolis

User project templates location: C:\Users\lux mundi\Documents\Visual Studio 10\Templates\ProjectT

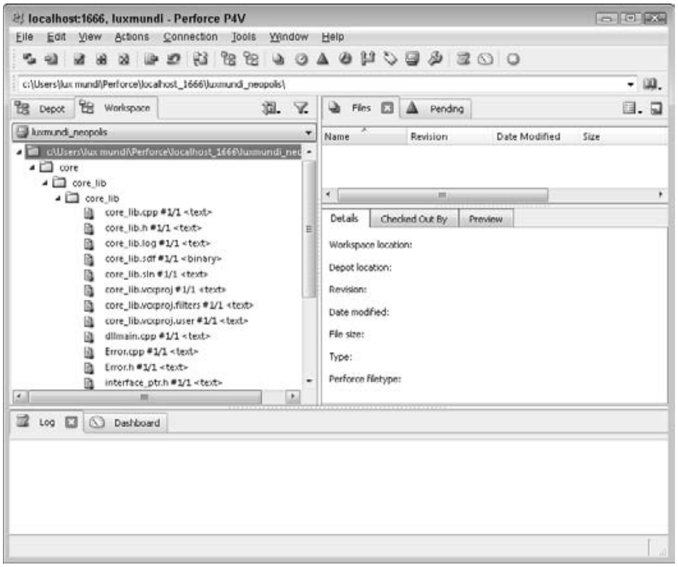
User item templates location: C:\Users\lux mundi\Documents\Visual Studio 10\Templates\ItemTen

☑ Always show Error List if build finishes with errors
☑ Track Active Item in Solution Explorer
☑ Show advanced build configurations
☑ Always show solution
☑ Save new projects when created
☑ Warn user when the project location is not trusted
☑ Show Output window when build starts
☑ Prompt for symbolic renaming when renaming files

Figure 3-1: You can set the default project location.

You must have Perforce installed on the client in order to use the Source Control menu in Visual Studio.

The working folder appears in Perforce, as shown in Figure 3-2.



c:\Users\lux mundi\Perforce\localhost_1666\luxmundi_neopolis\

luxmundi_neopolis

- c:\Users\lux mundi\Perforce\localhost_1666\luxmundi_neo
  - core
    - core_lib
      - core_lib
        - core_lib.cpp #1/1 <text>
        - core_lib.h #1/1 <text>
        - core_lib.log #1/1 <text>
        - core_lib.sdf #1/1 <binary>
        - core_lib.sln #1/1 <text>
        - core_lib.vcxproj #1/1 <text>
        - core_lib.vcxproj.filters #1/1 <text>
        - core_lib.vcxproj.user #1/1 <text>
        - dllmain.cpp #1/1 <text>
        - Error.cpp #1/1 <text>
        - Error.h #1/1 <text>
        - interface_ptr.h #1/1 <text>

Workspace location:

Depot location:

Revision:

Date modified:

File size:

Type:

Perforce filetype:

Figure 3-2: You can select the working folder for Perforce.

If you don't set your local working folder in your source control provider, Visual Studio uses the default projects location to download source code. Use the local working folder if you want to download controlled source code to a different location than your default projects folder in Visual Studio.

Binding to a Source Control Provider

After you install your source code control provider, you can access the source code control features in Visual Studio. Promptly add your new or existing solutions and projects to your source code control database. The binding process creates a link between a local copy of source code and the source control database.

Visual Studio adds files to your source code control database in this way:

1. **Creates a parent folder in your database named `<solution>.root` where `<solution>` is the solution name in Visual Studio.**
2. **Adds a folder to the `*.root` folder for your solution.**
3. **Adds folders for each project to the solution folder.**

Microsoft recommends that you use solutions to organize your source code. You should add your solutions to source code control. When you add a solution, all its projects are added, too. If you add a new project to a solution, you can easily add that project to source code. Follow these rules for working with solutions and projects with source code control:

1. **Create a blank solution first.**
2. **Always add projects to solutions.**
3. **Name your solutions so that they identify the system under development.**

The following sections explain the steps in more detail.

Adding a blank solution

To create a blank solution, follow these steps:

1. **Press Ctrl+Shift+N to open the New Project window.**
2. **Click the plus sign (+) next to Other Project Types.**
3. **Select Visual Studio Solutions.**

   A list of solution types appears in the Templates pane.
4. **Select Blank Solution.**
5. **Type a name for the solution and then click OK.**

   Select the Add to Source Control option to add the solution to source control when it's created.

Adding an existing solution

Here's how to add a solution to a Perforce source code control database by using Visual Studio:

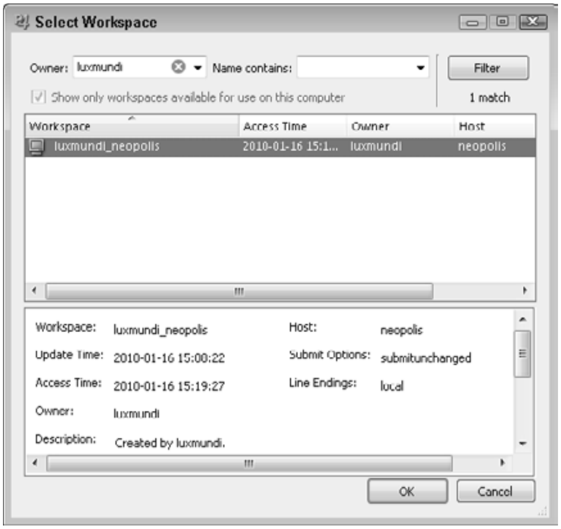
1. **Open or create the solution in Visual Studio.**
2. **Choose File➪Source Control.**

   A submenu appears.
3. **Select Add Solution to Source Control.**

   The Perforce login appears. At this step, Visual Studio passes off your request to your source code control provider.
4. **Type your login credentials to access the Perforce database and then click OK.**

   The Add to SourceSafe window appears.
5. **Select a work space in Perforce for your solution.**

   Visual Studio provides the default name `<*solutionname*>.root`. You should generally accept the default name, as shown in Figure 3-3.



Select Workspace

Owner: luxmundi

Name contains:

☑ Show only workspaces available for use on this computer

1 match

Workspace	Access Time	Owner	Host
luxmundi_neopolis	2010-01-16 15:1...	luxmundi	neopolis

Workspace: luxmundi_neopolis

Update Time: 2010-01-16 15:00:22

Access Time: 2010-01-16 15:19:27

Owner: luxmundi

Description: Created by luxmundi.

Host: neopolis

Submit Options: submitunchanged

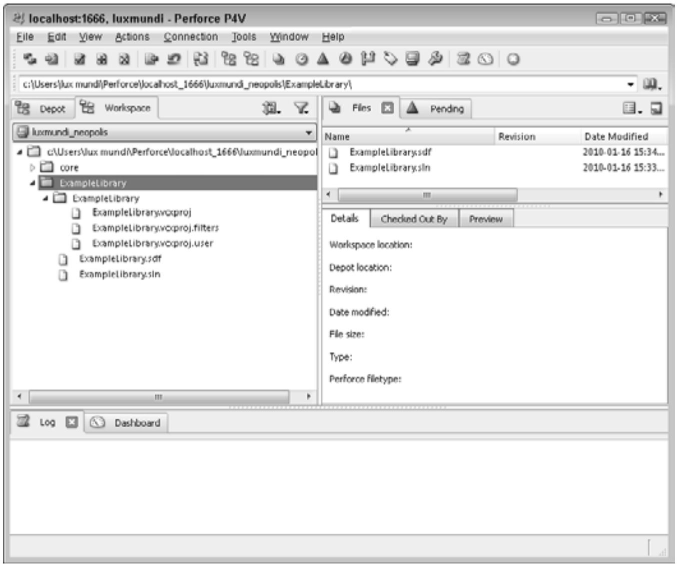
Line Endings: local

Figure 3-3: Select workspace in Perforce.

Ideally, you want your solution's name to describe the system you're developing. If the solution name is the same as the project name, distinguishing between the solution and project folders can be confusing.

6. **Click OK.**

   Figure 3-4 shows the new solution `ExampleLibrary` added to your Perforce workspace.


localhost:1666, luxmundi - Perforce P4V
File
Edit
View
Actions
Connection
Tools
Window
Help
c:\Users\lux mundi\Perforce\localhost_1666\luxmundi_neopolis\ExampleLibrary\
Depot
Workspace
luxmundi_neopolis
c:\Users\lux mundi\Perforce\localhost_1666\luxmundi_neopol
core
ExampleLibrary
ExampleLibrary
ExampleLibrary.vcxproj
ExampleLibrary.vcxproj.filters
ExampleLibrary.vcxproj.user
ExampleLibrary.sdf
ExampleLibrary.sln
Files
Pending
Details
Checked Out By
Preview
Log
Dashboard

Name	Revision	Date Modified
ExampleLibrary.sdf		2010-01-16 15:34...
ExampleLibrary.sln		2010-01-16 15:33...

Workspace location:

Depot location:

Revision:

Date modified:

File size:

Type:

Perforce filetype:

Figure 3-4: The new solution `ExampleLibrary` in the Perforce workspace.

TIP

Don't spend a lot of energy organizing your source control folders in your source control provider. In reality, you don't spend much time using the source control view of your solutions and projects. You mostly interact with Solution Explorer in Visual Studio. Instead, focus on using meaningful names in Solution Explorer and accept the defaults in your source control database.

Adding a project to an existing solution

Here's how to add a project to an existing solution already in the source code control database:

1. **Add the project to the source-controlled solution.**
2. **Choose File➪Source Control.**

   A submenu appears.
3. **Choose Add Selected Projects to Source Control.**

   A dialog box prompts you to confirm your choice.

4. **Click Yes.**

   Perforce adds your project to the database under the existing solution's root folder.

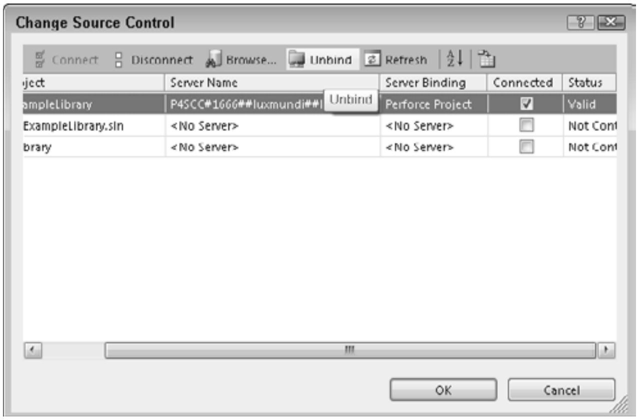
Breaking a binding

You may want to unbind source code from the source code control database — say, when you're working with developers (such as contractors who don't have access to Perforce) or when you no longer want the file included in the project. To unbind your local copy from the database, follow these steps:

1. **Open the solution or project under source control in Visual Studio.**
2. **Choose File⇨Source Control.**

   A submenu appears.
3. **Choose Change Source Control.**

   Use the Change Source Control window to view the binding properties, such as server name, database, and local working folder. Click the Columns button to add columns to the view.
4. **Click the solution or project to unbind; then click the Unbind button, as shown in Figure 3-5.**



ject	Server Name	Server Binding	Connected	Status
ampleLibrary	P4SCC#1666##luxmundi##l	Perforce Project	☑	Valid
ExampleLibrary.sln	<No Server>	<No Server>	□	Not Con
brary	<No Server>	<No Server>	□	Not Con

Figure 3-5: Unbind source code from the database from the Change Source Control dialog box.

Visual Studio prompts you to confirm your choice.

The binding is removed.

The source code remains in the database. Only the link between the local copy and the database is severed.

You can use the Change Source Control to connect and disconnect from the source control database. A *connection* is a live data link to the database. Disconnecting allows you to work offline from the database. You can connect to the database when you have a network connection again.

Performing Common Source Control Tasks

Source control bound to a source control database is locked. To edit a source code file that's under source code control, you need to first check it out. After you complete your edit, you check the file back into the source control database. Any time you work with source code, you should always synchronize your local copy of the source code with the master copy in the source control database.

The number of source control tasks you use daily is usually limited to three:

1. Get the latest version of source code.
2. Check the source code into the database.
3. Check the source code out of the database.

In Visual Studio, you select your source control provider by choosing File➪Source Control. You're likely to manage your source code in Visual Studio in these ways:

- **Right-click an item in Solution Explorer.**
- **Use the Source Control toolbar.** Choose View➪Toolbars➪Source Control to open the Source Control toolbar, shown in Figure 3-6.

Figure 3-6: The Source Control toolbar makes tasks easier.

- **Use the Pending Check-Ins window to manage checked-out source control.** To get to the Pending Check-Ins window, right-click the tree node of the source module you want to check in.

Table 3-1 describes common source control commands.

Table 3-1 Common Source Control Commands

Command	*What It Does*
Check In	Checks a file into the source control database
Check Out for Edit	Checks out a file from the source control database so that it can be edited
Compare	Shows the differences between two files
Get	Creates a read-only copy of the file in the local working folder
Get Latest	Gets the latest version of the file
Undo Checkout	Discards the last checkout
View History	Displays a file's history

Visual Studio provides visual clues about a file's status — whether it's checked in or checked out, or not source controlled. A visual cue is displayed next to certain items in Solution Explorer:

- **Checked-in items**: A lock
- **Checked-out items:** A check mark
- **Items not in source control**: A plus sign (+)

Figure 3-7 shows the visual cues in Solution Explorer.

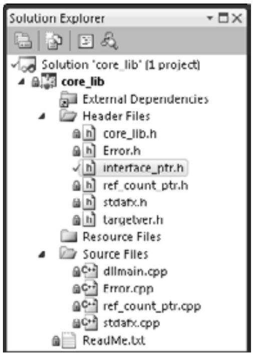


Figure 3-7: Locks and check marks indicate checked-out and checked-in source code.

Retrieving files from source control

The Get and Get Latest Version Visual Studio commands retrieve files from source control. You use these commands with a solution or project that's already under source control. See the earlier section "Binding to a Source Control Provider" to see how to add a solution or project to source control.

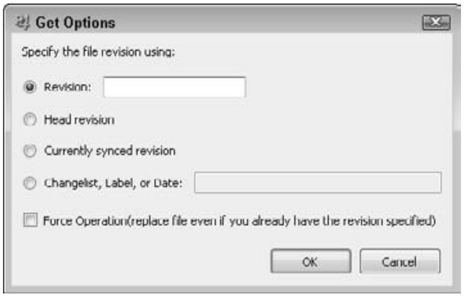
Source control providers "version" your source control files each time you perform a check-in. In most cases, you use the Get Latest Version command to retrieve the most recently checked-in version of a file. You use the Get command if you want to specify additional options, such as whether to make the file writeable when it's retrieved.

You use the View History command along with Get to access previous versions of projects and files. To open a previous version of a file, follow these steps:

1. **Right-click the file in the Solution Explorer for which you want to retrieve a version.**
2. **Choose Get from the shortcut menu.**

   The Get window appears.
3. **Click the Options button at the left side of the toolbar.**

   The Get Options window appears, as shown in Figure 3-8.



Get Options

Specify the file revision using:

● Revision:

○ Head revision

○ Currently synced revision

○ Changelist, Label, or Date:

□ Force Operation(replace file even if you already have the revision specified)

Figure 3-8: Use Get Options to get a particular version of a file.

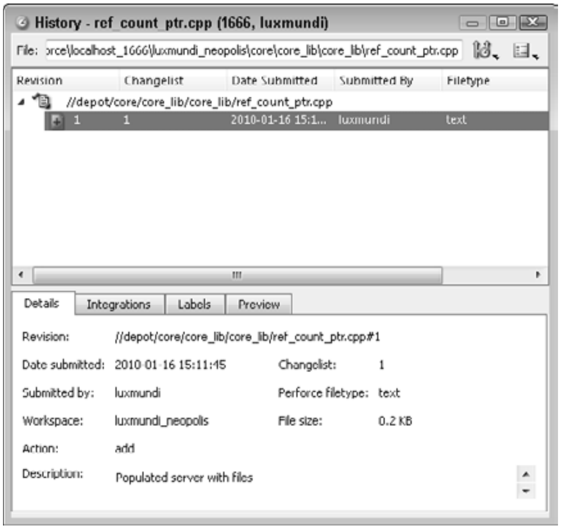
4. **Choose one of the options from Figure 3-8 and click OK.**

   The Get Options dialog box closes.
5. **Click OK.**

   The Get Window closes and Visual Studio retrieves the file.

To view a file history from Visual Studio:

1. **Right-click the file in Solution Explorer.**
2. **Choose View History from the contextual menu.**
3. **The History dialog box appears as shown in Figure 3-9.**



File: orce\localhost_1666\luxmundi_neopolis\core\core_lib\core_lib\ref_count_ptr.cpp

Revision	Changelist	Date Submitted	Submitted By	Filetype
//depot/core/core_lib/core_lib/ref_count_ptr.cpp				
1	1	2010-01-16 15:1...	luxmundi	text

Revision: //depot/core/core_lib/core_lib/ref_count_ptr.cpp#1

Date submitted: 2010-01-16 15:11:45 Changelist: 1

Submitted by: luxmundi Perforce filetype: text

Workspace: luxmundi_neopolis File size: 0.2 KB

Action: add

Description: Populated server with files

Figure 3-9: The History window shows you versions.

Editing source-controlled files

You must check out source-controlled files before you can edit them. You can check out a single file or an entire project. Visual Studio provides a Check Out for Edit command that instructs your source control provider to

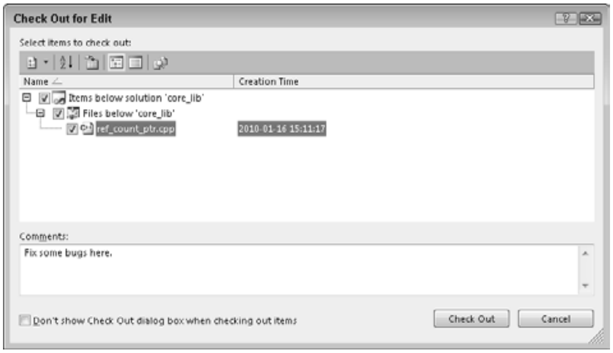
- Download the latest version of the source-controlled file
- Mark the file as writeable

To check out a file by using the Perforce plug-in, follow these steps:

1. **Right-click the file in Solution Explorer.**
2. **Choose Check Out for Edit from the shortcut menu.**

   The Check Out for Edit window displays a tree view of all the files that must be checked out. For example, the project file is automatically checked out when you check out a code file, as shown in Figure 3-10.
3. **(Optional) Type a comment to describe the reason for the checkout.**
4. **Click the Check Out button.**

   Perforce downloads the latest versions of the files to your local working folder and marks them as writeable. A check mark appears next to the filenames in Solution Explorer.



☑ Items below solution 'core_lib'
☑ Files below 'core_lib'
☑ ref_count_ptr.cpp 2010-01-16 15:11:17

Comments:
Fix some bugs here.

□ Don't show Check Out dialog box when checking out items

Figure 3-10: You see the hierarchy tree when you check out source code.

To check out a previous version of a source-controlled file, follow these steps:

1. **Download the version of the file you want from the source control database by following the steps in the preceding section.**
2. **Execute the Check Out for Edit command.**

   The Check Out for Edit window appears.
3. **Click the Options button (the first button) on the toolbar.**
4. **Select Check Out Local Version.**

   If the Check Out Local Version command is unavailable, click Advanced and then select the Don't Get Local Copy option.
5. **Click the Check Out button.**

   Perforce checks out the local copy of the file.

You can undo a checkout by using the Undo Checkout command. Any changes to the local file are lost.

Depending on how Perforce is deployed, it may place an exclusive lock on your file when it's checked out. This lock prevents anyone else from checking out the file from the database. If your database is configured to use multiple checkouts, other users can check out the file. As you may imagine, this lock is a potentially dangerous Perforce configuration. All changes are merged in the database. Check with your database administrator to determine how your source control provider is configured.

Multiple checkouts are often used with ASP.NET Web sites. Multiple developers, therefore, can check out the entire Web site. Because developers aren't usually working on exactly the same source file, merge issues are infrequent.

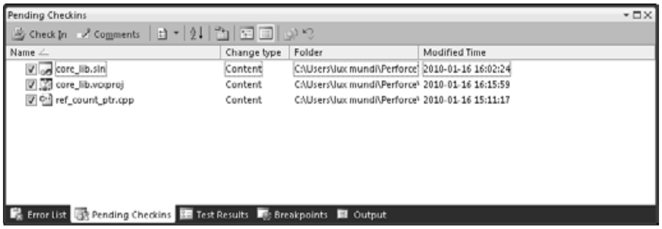
Checking files into source control

You can check in files by using the Check In command. You can use the Pending Checkins window in Visual Studio to display a list of all files that are checked out for the open solution.

To use the Check In window, follow these steps:

1. **Right-click any file in Solution Explorer.**
2. **Choose View Pending Checkins from the shortcut menu.**

   The Pending Checkins window appears, displaying checked-out files in tree view by default. Click the Flat View button to display a flat list of files, as shown in Figure 3-11.



Name	Change type	Folder	Modified Time
☑ core_lib.sln	Content	C:\Users\lux mundi\Perforce	2010-01-16 16:02:24
☑ core_lib.vcxproj	Content	C:\Users\lux mundi\Perforce	2010-01-16 16:15:59
☑ ref_count_ptr.cpp	Content	C:\Users\lux mundi\Perforce	2010-01-16 15:11:17

Figure 3-11: The Pending Checkins dialog box.

3. **Clear the check boxes for any files you don't want to check in now.**
4. **Click the Comments button (the fifth button from the left on the toolbar) to add a check-in comment to all the files.**
5. **Click the Check In button.**

   The files are checked in to source control and marked as Read-Only in the local working folder.

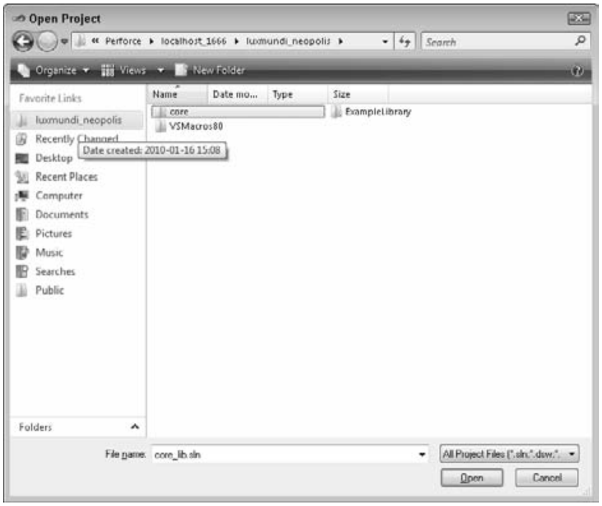
Retrieving a solution or project from source control

After a new solution or project is added to source code control, you must download the solution or project from source code control before you can use it. For example, if you're working in a team environment, you may need to retrieve a solution or project that another team member creates and adds to source control.

To open a project from the source control database, follow these steps:

1. **Press Ctrl+Shift+O to open the Open Project window.**
2. **Click the icon for your source control provider along the left side of the window.**

   A list of source control databases appears in the File Explorer. Figure 3-12 shows an example of accessing a Perforce workspace.



File name: core_lib.sln

All Project Files (*.sln;*.dsw;*.

Figure 3-12: Your solutions appear when you click the Perforce workspace icon.

3. **Navigate to your solution or project file and then click Open.**

   Visual Studio downloads the files from the source control database to the local working folder.

   See the earlier section "Getting ready for source control" for information on how to set the local working folder.

You can open a Web site from the source control database by using the Open Web Site window in Visual Studio.

Source Code Control in the Real World

Source code control is an activity that falls under a larger area of the software development life cycle known as software configuration management (SCM). SCM is often referred to as *change management* because managing the changes to software is one of SCM's primary goals.

Software configuration management encompasses many activities, such as

- Identifying items, such as source code, subject to change management
- Defining a change control process for managing changes
- Controlling software builds and releases

In practical terms, source code control is a key element in managing changes, builds, and releases. Here are some other tools that enable SCM:

- Requirements-tracking software
- Bug-tracking software
- Scripts for creating builds

By using source code control software in conjunction with other SCM tools, you can

- Trace software features to requirements.
- Trace code changes to bug-fix requests.
- Track versions.
- Reproduce builds.
- Create a traceable history of who did what and when.
- Identify differences among releases.

You can read about software configuration management in the Software Engineering Body of Knowledge (SWEBOK). The SWEBOK represents the latest thinking on how to engineer good software. Even if you don't believe that software should be engineered, you can get plenty of good information from the SWEBOK. You can download it at `www.swebok.org`.

A software project has many items, in addition to source code, that need to be controlled. Here are some examples, which are often referred to as *artifacts* or *work products:*

- Requirements specifications
- Architecture diagrams
- Build scripts
- Installer programs
- Compilers

Some people choose to place these artifacts in a source control tool, such as Perforce. Because some items, such as requirements documents, aren't source-controlled, many teams use document management systems, such as Microsoft Windows SharePoint Services or Confluence.

The Microsoft product Team Foundation Server (TFS) helps teams manage a project's artifacts and the SCM process. TFS works hand in hand with Visual Studio to create an integrated change-management process.

Regardless of which tools you use, be sure to follow these best practices when you use source control:

- Get the latest version of source code frequently.
- Check in your changes to source code frequently.
- Build your software from the master source code daily.
- Back up your master source code.
- Implement bug-tracking software that links to your source code changes.
- Utilize a code collaboration tool to ensure changes are peer reviewed before submission. A powerful, web-based code collaboration tool is SmartBear's Code Collaborator (`http://smartbear.com/codecollab.php`)
- Include everything you need to reproduce a build, including build scripts, installers, and compilers.

Going Beyond Integration

The daily integration of each developer's changes to source code is only one piece of the source control puzzle. Source code control providers, such as Perforce, support additional commands related to managing version control of your software, such as

- `Share`: Shares source code between two projects without using copy and paste.
- `Branch`: Splits off a file or project as a separate file or project. For example, if you branch a file at version 3, you have two independent copies of version 3.
- `Merge`: Combines the contents of two files and creates a new, merged version. You still have two files, but the contents are merged.
- `Diff`: Compares two versions of a file. You can use the `Merge` command to create a new merged version.

- `Label`: Associates a lookup tag with the source-controlled item. Labels are often used at the project level for version control. For example, the daily build process may use a label to associate a build number with a project. Team members can use the build number to retrieve the source code used to create the build from the database.
- `Rollback`: Erases all successive versions after a chosen version.
- `Pin`: Marks a version of a file that can't be changed until the pin is removed.

Visual Studio 2010 has menu access to many of these commands. However, in most shops, a source control administrator performs these tasks using whichever tool is most appropriate, which is often a batch file.

Perforce has a command line feature that you can use to execute source control commands on the command line or in a batch file. Perforce even has the ability for you to add custom commands to your environment to execute project-specific tasks such as integration with bug tracking systems and code review tools. One such application is to customize Perforce to require all check-ins to be peer reviewed. A custom Perforce command can send the files to a code collaboration system to be reviewed before submission to the Perforce database. Customizing Perforce is beyond the scope of this text, but you can see how source code control is a powerful tool in your software development arsenal.

Chapter 4: Writing Custom Facebook Applications

In This Chapter

- **Developing applications in Facebook**
- **Writing your custom application code**
- **Executing your first Facebook application**

Starting out as a networking site for Harvard University students, Facebook has exploded into a social media phenomenon that has swept nearly the entire world. Besides being a social networking site, Facebook has become a development platform for application developers. You can develop Facebook applications using several programming languages, such as PHP, Ruby on Rails, JavaScript, Python, C#, and Visual Basic. Your application can be either a desktop application or a Web application. In this chapter, you find out how to develop an ASP.NET web application for Facebook.

Creating and Registering Your Application with Facebook

Facebook (`www.facebook.com`) is a social media networking site launched in 2006 that allows anyone over the age of 13 with a valid e-mail address to create an account. At the time of this writing, Facebook had approximately 400 million users, a number that is growing rapidly.

After you have an account on Facebook, you can join any number of networks created according to geographical location, workplace, church affiliation, high school or college, and so on. Once on Facebook, you can search for classmates, coworkers, colleagues, or any other person you want to keep in touch with. To connect with another user, you simply ask that person to be your friend; if he accepts, you're now connected. After you're connected, you and your new friend can begin sharing messages and photos. You can also see who the friends of your friends are so that in no time you have developed a substantial social network of friends, coworkers, colleagues, and any other person that shares the same interests as you.

Applications such as games, quizzes, puzzles, and others have become popular on Facebook. The applications allow you to share information with others on Facebook as well as interact directly with other users while you are playing. Besides games, you can create any other kind of application and utilize Facebook information to enhance the experience of that application. Say, for example, you have a mapping application where users can enter someone's location and see on a map where that user lives. With Facebook, you can get information for all your friends and place a pin on the map for each user's location. So now at a glance you can see where all of your Facebook friends are living.

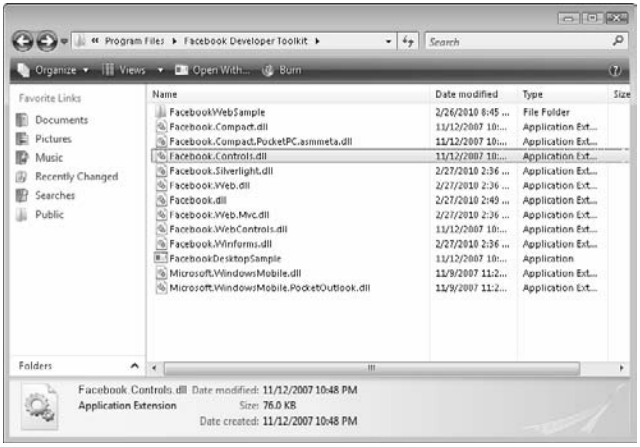
Facebook allows any registered user to develop applications for Facebook. If you don't have an account on Facebook, you can create one for free at `www.facebook.com`. Before your code can use the Facebook platform, you first have to create and register the application with Facebook. Facebook requires you to give them some information about your application and also your contact information so they may contact you about your application if necessary.

To create an application on Facebook, you go to `http://developers.facebook.com/get_started.php`. The Facebook developers page opens. Here, you find all the information needed to get you started with your application, including a detailed description of configuration options for your application. Most importantly, you need an `ApplicationPrivateKey` and `ApplicationSecret` for your application to connect to Facebook. The `ApplicationPrivateKey` and `ApplicationSecret` let Facebook know about your application and that it has authorized access to Facebook information.

Creating Your Application with Visual Studio 2010

Facebook provides a developer Software Development Kit (SDK) for application development. The current version is 3.0, and you can download this SDK from the CodePlex site at `http://facebooktoolkit.codeplex.com`. Once downloaded, you can extract the SDK to any directory you choose. After it's installed, your SDK directory looks something like Figure 4-1.

In this example, you create an ASP.Net application that provides the current weather forecast for the logged-in user, as well as the current weather forecast for a friend's current location. The weather forecast data come from Google. You can find more details about accessing weather data from Google in Book IV, Chapter 9.



Name	Date modified	Type	Size
FacebookWebSample	2/26/2010 8:45 ...	File Folder	
Facebook.Compact.dll	11/12/2007 10:...	Application Ext...	
Facebook.Compact.PocketPC.asmmeta.dll	11/12/2007 10:...	Application Ext...	
Facebook.Controls.dll	11/12/2007 10:...	Application Ext...	
Facebook.Silverlight.dll	2/27/2010 2:36 ...	Application Ext...	
Facebook.Web.dll	2/27/2010 2:36 ...	Application Ext...	
Facebook.dll	2/27/2010 2:49 ...	Application Ext...	
Facebook.Web.Mvc.dll	2/27/2010 2:36 ...	Application Ext...	
Facebook.WebControls.dll	11/12/2007 10:...	Application Ext...	
Facebook.Winforms.dll	2/27/2010 2:36 ...	Application Ext...	
FacebookDesktopSample	11/12/2007 10:...	Application	
Microsoft.WindowsMobile.dll	11/9/2007 11:2...	Application Ext...	
Microsoft.WindowsMobile.PocketOutlook.dll	11/9/2007 11:2...	Application Ext...	

Facebook.Controls.dll Date modified: 11/12/2007 10:48 PM
Application Extension Size: 76.0 KB
Date created: 11/12/2007 10:48 PM

Figure 4-1: Files installed for the Facebook SDK.

You can download the complete program for this example from this book's companion Web site at `www.dummies.com/go/vs2010`.

Creating a Facebook application is as simple as creating any other type of application with Visual Studio. To create an ASP.NET Facebook Web application:

1. **Choose File⇨New⇨Project and choose the ASP.Net Web Application from the list of project templates under Visual C#.**

   Visual Studio creates a new C# ASP.Net Web Application.

2. **From the Solution Explorer, right click the References node and choose Add New Reference from the context menu.**

   The File Open dialog box appears.

3. **Navigate to the directory where you installed the Facebook SDK and choose the files Facebook.dll and Facebook.Web.dll.**

   You should see these references in the Solution Explorer, as shown in Figure 4-2.

The Facebook SDK supports Web, desktop, and mobile development. The `Facebook.dll` contains the main classes used for the Facebook SDK. The `Facebook.Web.dll` adds classes that developers use to create ASP. Net Web applications. There is also `Facebook.Silverlight.dll` for Silverlight applications and `Facebook.Winforms.dll` for Windows Forms desktop application development.

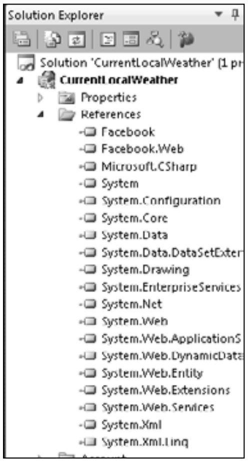


Figure 4-2: Solution Explorer shows the references for the Facebook SDK.

Writing your application code

After you have an ASP.Net project created and the necessary Facebook assemblies referenced in the project, you can get to work writing your application code.

To begin, you need to provide Facebook with the `ApplicationPrivateKey` and `ApplicationSecret` codes you obtained when you created and registered your application with Facebook. Without these keys, your application won't be able to connect to Facebook. A simple way of providing these keys to Facebook is to put them in your `web.config` file:

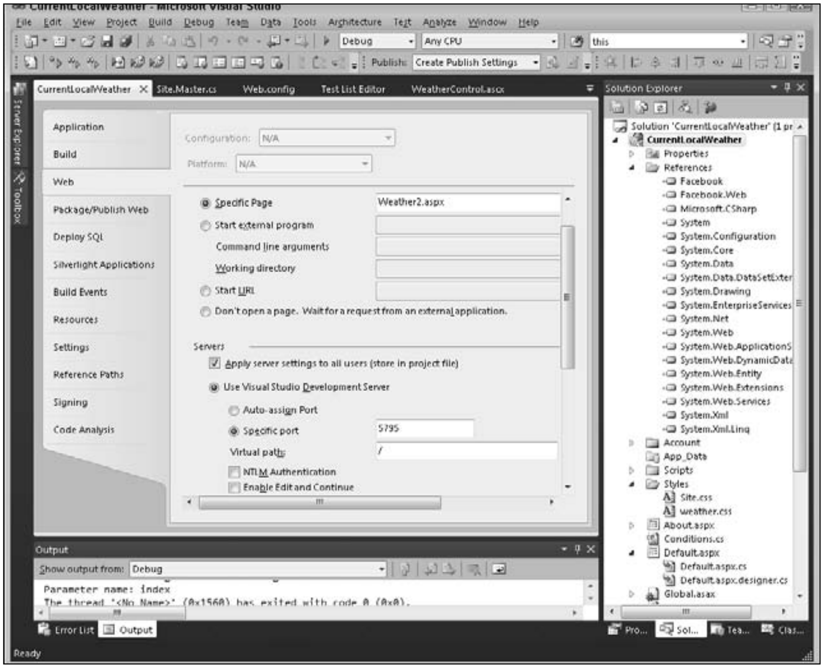
```
<appSettings>
    <add key="APIKey" value="Your Application Key"/>
    <add key="Secret" value="Your Application Secret"/>
    <add key="Callback" value=" http://localhost:5795/Weather2.aspx"/>
    <add key="Suffix" value="mylocalweather"/>
  </appSettings>
```

Don't share your application key or application secret with those who aren't developing your application. Doing so allows unscrupulous application developers to impersonate your application on Facebook, possibly causing your account to be revoked and damaging your reputation.

The `Callback` and `Suffix` keys tell Facebook how to call back to your application. Users access your application though Facebook, which then uses the `Callback` key to know which page of your application to load. The `Suffix` key is the suffix you want Facebook to use in the URL to access your application. In this example, the suffix is `mylocalweather`. When users access your application on Facebook it uses the URL `http://apps.facebook.com/mylocalweather` to load your application.

Facebook then calls into this application by loading the page specified in the Callback key, and your application executes and serves the resulting HTML to Facebook.

To ensure that your application can run in Facebook when launching from Visual Studio 2010, you need to set your project to execute under the same URL and load the page that you provided to Facebook in the Callback key in `web.config`. Your project settings should look like Figure 4-3.



Configuration: N/A
Platform: N/A

● Specific Page Weather2.aspx
○ Start external program
Command line arguments
Working directory
○ Start URL
○ Don't open a page. Wait for a request from an external application.

Servers
☑ Apply server settings to all users (store in project file)
● Use Visual Studio Development Server
○ Auto-assign Port
● Specific port 5795
Virtual path: /
☐ NTLM Authentication
☐ Enable Edit and Continue

Show output from: Debug

```
Parameter name: index
The thread '<No Name>' (0x1560) has exited with code 0 (0x0).
```

Figure 4-3: Project settings match specific page and specific port settings.

Connecting your application to Facebook

On Facebook, you can create two types of Web applications: IFrames and Facebook Markup Language (FBML). FBML lets you quickly start building an application from scratch and has fewer moving parts making it easy for the beginning developer. IFrames applications are easier and faster if you have an existing application or widget developed. IFrames applications are also faster than FBML applications and allow you to use JavaScript, HTML, and CSS as well as popular JavaScript libraries like jQuery. You can find more information at `http://wiki.developers.facebook.com/index.php/Choosing_between_an_FBML_or_IFrame_Application`.

This example is an IFrames Facebook application. To allow your application to connect to Facebook, modify your Site.master.cs code file to have your master page class derive from Facebook.Web.CanvasIFrameMasterPage:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace CurrentLocalWeather
{
    public partial class SiteMaster : Facebook.Web.CanvasIFrameMasterPage
    {
        public SiteMaster()
        {
            RequireLogin = true;
        }
        protected void Page_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Set RequireLogin to true so that your application requires users to be logged in for your application to access Facebook information. Without being logged in, your application will be limited on the information it can access.

Now your application has all it needs to connect to Facebook. The CanvasIFrameMasterPage class contains all the plumbing necessary for connecting to and authenticating your application on Facebook.

Laying out your application

After your application is connected to Facebook, you'll want it to do some real work. First, you need a page to display to the user when Facebook loads your applicationChoose Project➪Add New Item➪Web Form using Master Page and call the Master Page Weather2.aspx.

Next, to create a reusable Web User Control that does the work of accessing and displaying the weather data from Google, choose Project➪Add New Item, select Web User Control, and call it WeatherControl.ascx. This control contains text labels and image server controls to display forecast data and images associated with the forecast information.

You also can create style sheets to control the rendering of the HTML for this Web User Control. The following listings show the HTML and CSS code to render the Web User Control.

CSS:

```
.current_container
{
    width: 350px;
    margin-left: auto;
    margin-right: auto;
    margin:0;
    font-family: Segoe UI Arial Lucida Sans Unicode Sans-Serif;
}
.current
{
   text-align: center;
   width: 100px;
   margin-left: auto;
   margin-right: auto;
}
.city
{
    text-align: center;
    width: 100px;
    margin-left: auto;
    margin-right: auto;
}

.thumbnail
{
    float: left;
    width: 60px;
    border: 1px solid #999;
    margin: 0 15px 15px 0;
    text-align: center;
}
.clearboth
{
    clear: both;
}
```

.ascx code:

```
<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="WeatherControl.ascx.
    cs" Inherits="CurrentLocalWeather.WeatherControl" %>

<div class="thumbnail">
       <asp:Image ID="CurrentProfileImage" runat="server" Height="60px"
    Width="60px" /><br />

       <asp:Label ID="CurrentProfileName" runat="server" Text="" Font-
    Names="Segoe UI"></asp:Label>
   </div>

   <div class="current_container">
```

```
<div class="current">
  <asp:Image ID="CurrentConditionsImage" runat="server" Height="60px"
 Width="60px" /><br />

 <asp:Label ID="CurrentTempLabel" runat="server" Text="Current Temp"
        Font-Names="Segoe UI"></asp:Label><br />
</div>
<div class="city">
<asp:Label ID="City" runat="server" Text="City" Font-Names="Segoe UI"></
 asp:Label>
</div>
<br />


<div class="thumbnail">
<asp:Label ID="ForecastLabel1" runat="server" Text="Label" Font-Names="Segoe
 UI"></asp:Label><br />

    <asp:Image ID="ForecastImage1" runat="server" Height="60px" Width="60px"
 /><br />

<asp:Label ID="ForecastTemp1" runat="server" Text="Min/Max" Font-Names="Segoe
 UI"></asp:Label>

</div>

<div class="thumbnail">
  <asp:Label ID="ForecastLabel2" runat="server" Text="Label" Font-Names="Segoe
 UI"></asp:Label><br />

      <asp:Image ID="ForecastImage2" runat="server" Height="60px" Width="60px"
 /><br />

<asp:Label ID="ForecastTemp2" runat="server" Text="Min/Max" Font-Names="Segoe
 UI"></asp:Label>

</div>

<div class="thumbnail">
   <asp:Label ID="ForecastLabel3" runat="server" Text="Label" Font-
 Names="Segoe UI"
       ></asp:Label><br />

<asp:Image ID="ForecastImage3" runat="server" Height="60px" Width="60px" /><br
 />

  <asp:Label ID="ForecastTemp3" runat="server" Text="Min/Max" Font-
 Names="Segoe UI"></asp:Label>

</div>

<div class="thumbnail">
  <asp:Label ID="ForecastLabel4" runat="server" Text="Label" Font-Names="Segoe
 UI"
        ></asp:Label><br />

   <asp:Image ID="ForecastImage4" runat="server" Height="60px" Width="60px"
 /><br />
```

```
    <asp:Label ID="ForecastTemp4" runat="server" Text="Min/Max" Font-Names="Segoe
    UI"></asp:Label>

   </div>

   </div>

   <div class="spacer">
       &nbsp;
   </div>

<asp:Label ID="Label1" runat="server" Text="Select Friend:"></asp:Label>

   <asp:DropDownList ID="DropDownFriendList" runat="server" AutoPostBack="True"
    Font-Names="Segoe UI"
    onselectedindexchanged="DropDownFriendList_SelectedIndexChanged">
</asp:DropDownList>
```

After you insert your server controls and define your layout and styles, you can start coding to make it all work. In the code behind file WeatherControl.ascx.cs, you write the C# code that accesses the weather data and populates the server controls.

Querying Data from Facebook

The CanvasIFrameMasterPage class in Facebook.Web.dll contains all the properties and methods that you need to access Facebook data. The main class that you use in the Facebook SDK is Facebook.Rest.Api. This class is the window into the world of Facebook development. If you created the Weather2.aspx file to use the master page (see “Laying out your application”), accessing this class from code is as easy as the following line:

```
this.Master.Api
```

This class provides many methods and properties for accessing Facebook data, but perhaps the easiest and most efficient is through the use of the Facebook Query Language (FQL). You can find out more about FQL from http://wiki.developers.facebook.com/index.php/FQL. FQL uses a Structured Query Language (SQL) syntax to access data from the Facebook databases and is more efficient because the SDK allows you to batch queries into a MultiQuery object for execution. Executing multiple queries at once saves round-trip accesses to Facebook servers from across the network. In the PageLoad event of Weather2.aspx , you can get the profile information of the logged-in user as well as information about friends of the logged in user as follows:

```
protected void Page_Load(object sender, EventArgs e)
        {
            System.Console.WriteLine(this.Master.Api.Session.UserId);

            this.Master.Api.Connect.Session.Login();

            long userId = this.Master.Api.Users.GetLoggedInUser();

            Dictionary<string, string> multiQuery = new Dictionary<string,
    string>();

            multiQuery.Add("query1", "SELECT name, current_location,pic_square
    from user WHERE uid=" + userId);
            multiQuery.Add("query2", "SELECT name, current_location, pic_square,
    uid from user WHERE uid IN (SELECT uid2 from friend WHERE uid1="+ userId +
    ")");

            foreach (Facebook.Schema.fql_result fqlResult in this.Master.Api.Fql.
    Multiquery(multiQuery))
            {
                if (fqlResult.name == "query1")
                {
                    this.OnUserQueryComplete(fqlResult.fql_result_set.
    ToString());
                }
                else if (fqlResult.name == "query2")
                {
                    this.OnFriendQueryComplete(fqlResult.fql_result_set.
    ToString());
                }
            }
        }
```

The preceding code deserves some explanation. When Facebook loads your `Weather2.aspx` page, it uses the session key information for the user that is currently logged into Facebook on the computer in which the page is executing. Calling `GetLoggedInUser` returns the User Id for that user, which is a numeric value. After you have the User Id, you can query the Facebook database for information about that user. In this case, two queries are required. The first query queries information from the logged-in user's profile. In particular, the query retrieves the name, `current_location`, `pic_square` columns from the user table. The following FQL query retrieves user information:

```
"SELECT name, current_location,pic_square from user WHERE uid=" + userId;
```

Finding out about your friends

The second query retrieves the name, `current_location`, `pic_square`, and `uid` columns from the user table for each user that is a friend of the logged in user. You query the list of friends from the `uid2` column of the friend table in the Facebook database. The FQL query is a little more complex:

```
"SELECT name, current_location, pic_square, uid from user WHERE uid IN (SELECT
    uid2 from friend WHERE uid1="+ userId + ")"
```

The `Multiquery` method of the `Facebook.Rest.Fql` class batches both queries and sends them to the Facebook server. The Facebook server returns a result set for each of the queries. The FQL APIs return query results in XML format for storage in a `DataSet` object or for manual parsing. After the query results are retrieved, you can populate the user and friend data on the `WeatherControl` Web User Control as follows:

```
public class ProfileInfo
    {
        private string _City;
        private string _State;
        private string _PostalCode;
        private string _ProfileImage;
        private string _Name;


        public string City
        {
            get
            {
                return (_City);
            }
            set
            {
                _City = value;
            }
        }

        public string State
        {
            get
            {
                return (_State);
            }
            set
            {
                _State = value;
            }
        }

        public string PostalCode
        {
            get
            {
                return (_PostalCode);
            }
            set
            {
                _PostalCode = value;
            }
        }
        public string ProfileImage
        {
            get
            {
                return (_ProfileImage);
            }
            set
            {
                _ProfileImage = value;
            }
        }

        public string Name
        {
            get
            {
                return (_Name);
            }
            set
            {
                _Name = value;
            }
        }
    }

private void OnFriendQueryComplete(string result)
        {
            XmlDocument xmlDoc = new XmlDocument();

            xmlDoc.LoadXml(result);

            XmlNodeList userNodes = xmlDoc.GetElementsByTagName("user");

            List<ProfileInfo> friends = new List<ProfileInfo>();

            foreach (XmlNode node in userNodes)
            {
                ProfileInfo friend = new ProfileInfo();

                foreach (XmlNode userNode in node.ChildNodes)
                {

                    if (userNode.Name == "pic_square")
                    {
                        friend.ProfileImage = userNode.InnerText;
                    }
                    else if (userNode.Name == "name")
                    {
                        friend.Name = userNode.InnerText;
                    }
                    else if (userNode.Name == "current_location")
                    {
                        foreach (XmlNode locationNode in userNode.ChildNodes)
                        {
                            if (locationNode.Name == "city")
                            {
                                friend.City = locationNode.InnerText;
                            }
                    else if (locationNode.Name == "state")
                    {
                        friend.State = locationNode.InnerText;
                    }
                    else if (locationNode.Name == "zip")
                    {
                        friend.PostalCode = locationNode.InnerText;
                    }
                }
            }
        }
        friends.Add(friend);
    }

    this.MyWeather.Friends = friends;
}

private void OnUserQueryComplete(string result)
{
    XmlDocument xmlDoc = new XmlDocument();
    string strUser = string.Empty;
    string strImageUrl = string.Empty;
    xmlDoc.LoadXml(result);

    XmlNodeList imageNodes = xmlDoc.GetElementsByTagName("pic_square");

    if (imageNodes != null && imageNodes.Count > 0)
    {
        strImageUrl = imageNodes[0].InnerText;
    }

    XmlNodeList userNodes = xmlDoc.GetElementsByTagName("name");

    if (userNodes != null && userNodes.Count > 0)
    {
        strUser = userNodes[0].InnerText;
    }

    XmlNodeList nodes = xmlDoc.GetElementsByTagName("current_location");

    string strCity = string.Empty;
    string strState = string.Empty;
    string strZip = string.Empty;

    foreach (XmlNode node in nodes)
    {
        foreach (XmlNode locationNode in node.ChildNodes)
        {
            if (locationNode.Name == "city")
            {
                strCity = locationNode.InnerText;
            }
            else if (locationNode.Name == "zip")
            {
                strZip = locationNode.InnerText;
            }
            else if (locationNode.Name == "state")
            {
                strState = locationNode.InnerText;
            }
        }
    }
```

```
            if (!string.IsNullOrEmpty(strZip))
            {
                string strLocation = string.Empty;

                if (!string.IsNullOrEmpty(strCity))
                {
                    strLocation = strCity;
                }
                if (!string.IsNullOrEmpty(strState))
                {
                    strLocation += ("," + strState);
                }

                this.MyWeather.CurrentLocation = strLocation;
                this.MyWeather.ProfileUser = strUser;
                this.MyWeather.ProfileUserImage = strImageUrl;
            }
        }
```

Accessing weather data for your friend's location

After you have the profile information containing user and location information for the logged-in user as well as that user's friends, you can get weather information for their location in the `WeatherControl.ascx.cs` code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Net;
using System.Text;
using System.IO;
using System.Xml;

namespace CurrentLocalWeather
{
    public partial class WeatherControl : System.Web.UI.UserControl
    {
        private string _Url = "http://www.google.com/ig/api?weather=";
        private string _BaseUrl = "http://www.google.com";

        private CurrentConditions _CurrentConditions = new CurrentConditions();
        private List<ForecastConditions> _ForecastConditions = new
    List<ForecastConditions>();
        private ForecastInformation _ForecastInformation = new
    ForecastInformation();
        private string _CurrentLocation;
        private string _ProfileUser;
        private string _ProfileUserImage;

        private List<ProfileInfo> _Friends;

        public List<ProfileInfo> Friends
        {
            get
            {
                return (_Friends);
            }
    set
    {
        _Friends = value;
    }
}

public string CurrentLocation
{
    get
    {
        return (_CurrentLocation);
    }
    set
    {
        this._CurrentLocation = value;
    }
}

public string ProfileUser
{
    get
    {
        return (_ProfileUser);
    }
    set
    {
        _ProfileUser = value;
    }
}

public string ProfileUserImage
{
    get
    {
        return (_ProfileUserImage);
    }
    set
    {
        _ProfileUserImage = value;
    }
}

protected void Page_Load(object sender, EventArgs e)
{
    if (!this.IsPostBack)
    {
        this.DropDownFriendList.DataSource = this._Friends;
        this.DropDownFriendList.DataTextField = "Name";

        this.DropDownFriendList.DataBind();

        string actualUrl = _Url + _CurrentLocation;

        this.LoadForecastData(actualUrl);

    }
    else
    {
        string actualUrl = _Url + _CurrentLocation;

        this.LoadForecastData(actualUrl);
        }
    }

    private void OnResponseReady(IAsyncResult result)
    {
        HttpWebRequest request = (HttpWebRequest)result.AsyncState;

        HttpWebResponse response = (HttpWebResponse)request.
EndGetResponse(result);

        Encoding utf8Encoding = Encoding.UTF8;

        StreamReader responseStream = new StreamReader(response.
GetResponseStream(), utf8Encoding);
        string content = responseStream.ReadToEnd();

        this._ForecastConditions.Clear();

        BuildForecastFromXml(content);

        UpdateDisplay();
    }

    private void BuildForecastFromXml(string xmlContent)
    {
        XmlDocument xmlDoc = new XmlDocument();

        xmlDoc.LoadXml(xmlContent);

        XmlNodeList forecastNodes = xmlDoc.GetElementsByTagName("forecast_
information");

        foreach (XmlNode node in forecastNodes)
        {
            foreach (XmlNode forecastNode in node.ChildNodes)
            {
                XmlAttributeCollection attrs = forecastNode.Attributes;

                if (forecastNode.Name == "city")
                {
                    this._ForecastInformation.City = attrs.
GetNamedItem("data").InnerText;
                }
                else if (forecastNode.Name == "postal_code")
                {
                    this._ForecastInformation.PostalCode = attrs.
GetNamedItem("data").InnerText;
                }
                else if (forecastNode.Name == "forecast_date")
                {
                    this._ForecastInformation.ForecastDate = attrs.
GetNamedItem("data").InnerText;
                }
                else if (forecastNode.Name == "current_date_time")
                {
                    this._ForecastInformation.CurrentDateTime = attrs.
GetNamedItem("data").InnerText;
                }
            }

        }
        XmlNodeList currentConditionsNodes = xmlDoc.
GetElementsByTagName("current_conditions");

        foreach (XmlNode node in currentConditionsNodes)
        {
            foreach (XmlNode currentNode in node.ChildNodes)
            {
                XmlAttributeCollection attrs = currentNode.Attributes;

                if (currentNode.Name == "condition")
                {
                    this._CurrentConditions.ConditionData = attrs.
GetNamedItem("data").InnerText;
                }
                else if (currentNode.Name == "temp_f")
                {
                    this._CurrentConditions.TempF = attrs.
GetNamedItem("data").InnerText;
                }
                else if (currentNode.Name == "temp_c")
                {
                    this._CurrentConditions.TempC = attrs.
GetNamedItem("data").InnerText;
                }
                else if (currentNode.Name == "humidity")
                {
                    this._CurrentConditions.HumidityData = attrs.
GetNamedItem("data").InnerText;
                }
                else if (currentNode.Name == "wind_condition")
                {
                    this._CurrentConditions.WindData = attrs.
GetNamedItem("data").InnerText;
                }
                else if (currentNode.Name == "icon")
                {
                    this._CurrentConditions.IconData = attrs.
GetNamedItem("data").InnerText;
                }

            }
        }

        XmlNodeList forecastDataNodes = xmlDoc.
GetElementsByTagName("forecast_conditions");

        foreach (XmlNode node in forecastDataNodes)
        {
            ForecastConditions cond = new ForecastConditions();

            foreach (XmlNode forecastDataNode in node.ChildNodes)
            {
                XmlAttributeCollection attrs = forecastDataNode.Attributes;

                if (forecastDataNode.Name == "day_of_week")
                {
                    cond.DayOfWeek = attrs.GetNamedItem("data").InnerText;
                }
                else if (forecastDataNode.Name == "low")
                {
                    cond.LowTemp = attrs.GetNamedItem("data").InnerText;
                }
                else if (forecastDataNode.Name == "high")
                {
                    cond.HighTemp = attrs.GetNamedItem("data").InnerText;
                }
                else if (forecastDataNode.Name == "icon")
                {
                    cond.IconData = attrs.GetNamedItem("data").InnerText;
                }
                else if (forecastDataNode.Name == "condition")
                {
                    cond.ConditionData = attrs.GetNamedItem("data").
InnerText;
                }
            }

            this._ForecastConditions.Add(cond);
        }

    }

    private void UpdateDisplay()
    {
        try
        {
            this.ForecastLabel1.Text = this._ForecastConditions[0].DayOfWeek;
            this.ForecastImage1.ImageUrl = this._BaseUrl + this._
ForecastConditions[0].IconData;

            this.ForecastLabel2.Text = this._ForecastConditions[1].DayOfWeek;
            this.ForecastImage2.ImageUrl = this._BaseUrl + this._
ForecastConditions[1].IconData;

            this.ForecastLabel3.Text = this._ForecastConditions[2].DayOfWeek;
            this.ForecastImage3.ImageUrl = this._BaseUrl + this._
ForecastConditions[2].IconData;

            this.ForecastLabel4.Text = this._ForecastConditions[3].DayOfWeek;
            this.ForecastImage4.ImageUrl = this._BaseUrl + this._
ForecastConditions[3].IconData;

            this.ForecastTemp1.Text = this._ForecastConditions[0].HighTemp +
"/" + this._ForecastConditions[0].LowTemp;
            this.ForecastTemp2.Text = this._ForecastConditions[1].HighTemp +
"/" + this._ForecastConditions[1].LowTemp;
            this.ForecastTemp3.Text = this._ForecastConditions[2].HighTemp +
"/" + this._ForecastConditions[2].LowTemp;
            this.ForecastTemp4.Text = this._ForecastConditions[3].HighTemp +
"/" + this._ForecastConditions[3].LowTemp;

            this.CurrentTempLabel.Text = this._CurrentConditions.TempF + " F
/ " + this._CurrentConditions.TempC +" C";
            this.CurrentConditionsImage.ImageUrl = _BaseUrl + this._
CurrentConditions.IconData;
            this.City.Text = this._ForecastInformation.City;

            this.CurrentProfileImage.ImageUrl = this._ProfileUserImage;
            this.CurrentProfileName.Text = this._ProfileUser;
        }
        catch (System.Exception ex)
        {
            System.Diagnostics.Debug.WriteLine(ex.Message);
        }
    }

    private void LoadForecastData(string url)
    {
        try
        {
            HttpWebRequest request =
                (HttpWebRequest)WebRequest.Create(url);

            HttpWebResponse response = (HttpWebResponse)request.
GetResponse();

            Encoding utf8Encoding = Encoding.UTF8;

            StreamReader responseStream = new StreamReader(response.
GetResponseStream(), utf8Encoding);
            string content = responseStream.ReadToEnd();

            this._ForecastConditions.Clear();

            BuildForecastFromXml(content);

            UpdateDisplay();

        }
        catch (System.Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }

    protected void DropDownFriendList_SelectedIndexChanged(object sender,
EventArgs e)
    {
        if (this.IsPostBack)
        {
            ListItem item = this.DropDownFriendList.Items[this.
DropDownFriendList.SelectedIndex];

            if (item != null)
            {
                string userId = item.Value;

                /*Now, find this user*/
                IEnumerable<ProfileInfo> profile =
                    from friend in _Friends where friend.Name == userId
select friend;

                foreach (ProfileInfo info in profile)
                {
                    _ProfileUser = info.Name;
                    _ProfileUserImage = info.ProfileImage;
```

```
                        if (!String.IsNullOrEmpty(info.PostalCode))
                        {
                            _CurrentLocation = info.PostalCode;
                        }
                        else
                        {
                            string strLocation = string.Empty;

                            if (!String.IsNullOrEmpty(info.City))
                            {
                                strLocation += info.City;
                            }
                            if (!String.IsNullOrEmpty(info.State))
                            {
                                strLocation += ("," + info.State);
                            }
                            _CurrentLocation = strLocation;
                        }
                    }
                }
                string actualUrl = _Url + _CurrentLocation;

                this.LoadForecastData(actualUrl);

            }
        }
    }
}
```

Finally, you add a reference to the `WeatherControl.ascx` WebUserControl to your `Weather2.aspx` page for rendering weather data as follows:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.
    Master" AutoEventWireup="true" CodeBehind="Weather2.aspx.cs"
    Inherits="CurrentLocalWeather.WebForm1" %>
<%@ Register TagPrefix="AndrewMoore" TagName="Weather" Src="WeatherControl.ascx"
    %>

<%@ MasterType VirtualPath="~/Site.Master" %>

<asp:Content ID="Content1" ContentPlaceHolderID="HeadContent" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
<AndrewMoore:Weather id="MyWeather" runat="server"/>
</asp:Content>
```

When the application is complete, you can select F5 to launch it from Visual Studio just like you would any other ASP.Net application. You'll see your application load on the Web site and then make requests to Facebook, which in turn will call back to your Web application and request the *canvas,* which is the `Weather2.aspx` code you created. This code executes on your Web site and returns HTML to Facebook as shown in Figure 4-4.

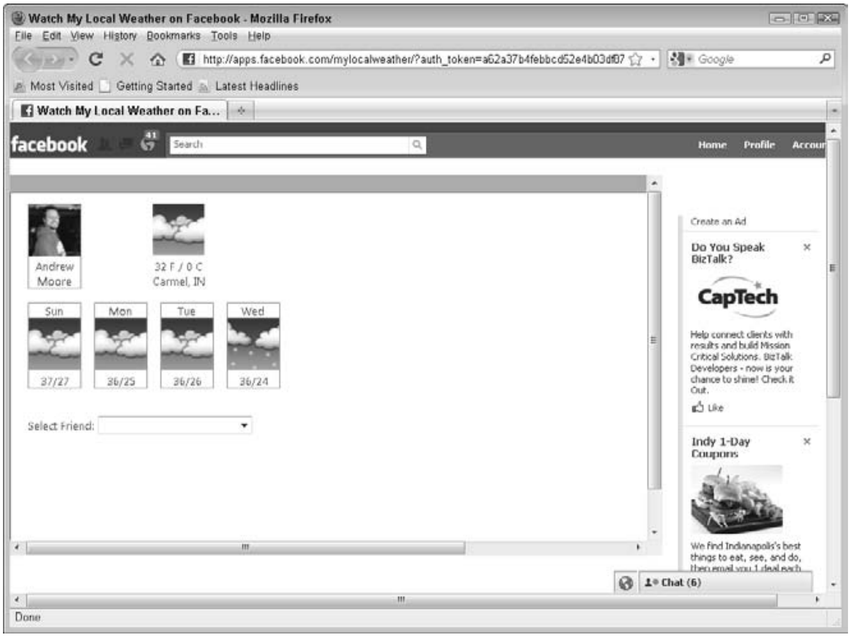
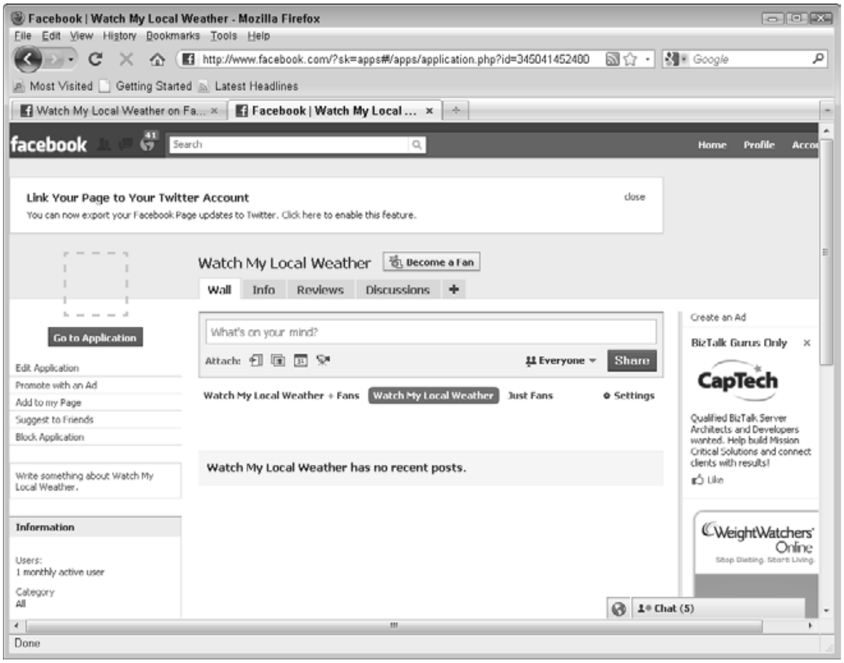

Watch My Local Weather on Facebook - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://apps.facebook.com/mylocalweather/?auth_token=a62a37b4febbcd52e4b03df07
Google
Most Visited Getting Started Latest Headlines
Watch My Local Weather on Fa...
facebook
41
Search
Home Profile Accour
Andrew Moore
32 F / 0 C
Carmel, IN
Sun
37/27
Mon
36/25
Tue
36/26
Wed
36/24
Select Friend:
Create an Ad
Do You Speak BizTalk?
CapTech
Help connect clients with results and build Mission Critical Solutions. BizTalk Developers - now is your chance to shine! Check it Out.
Like
Indy 1-Day Coupons
We find Indianapolis's best things to eat, see, and do, then email you 1 deal each
Chat (6)
Done

Figure 4-4: Your application executes and displays on Facebook.

Executing Your Application on Facebook

You can execute your application directly from Facebook by entering `http://apps.facebook.com/{my application name}`. After you're happy with your application, you can deploy it to a permanent Web site that is accessible by users of your application. Now you can tell all your friends about your application, they can tell their friends, and so on.

Facebook also tracks metrics about your application, such as the number of active users. From the Facebook Developer application, you can view this information and advertise and invite friends to use your application. Here, you find your application listed, and you can view or modify your application's profile and configuration settings. Your application profile resembles Figure 4-5.



http://www.facebook.com/?sk=apps#/apps/application.php?id=345041452480

Link Your Page to Your Twitter Account

You can now export your Facebook Page updates to Twitter. Click here to enable this feature.

Watch My Local Weather has no recent posts.

Write something about Watch My Local Weather.

Information

Users:
1 monthly active user

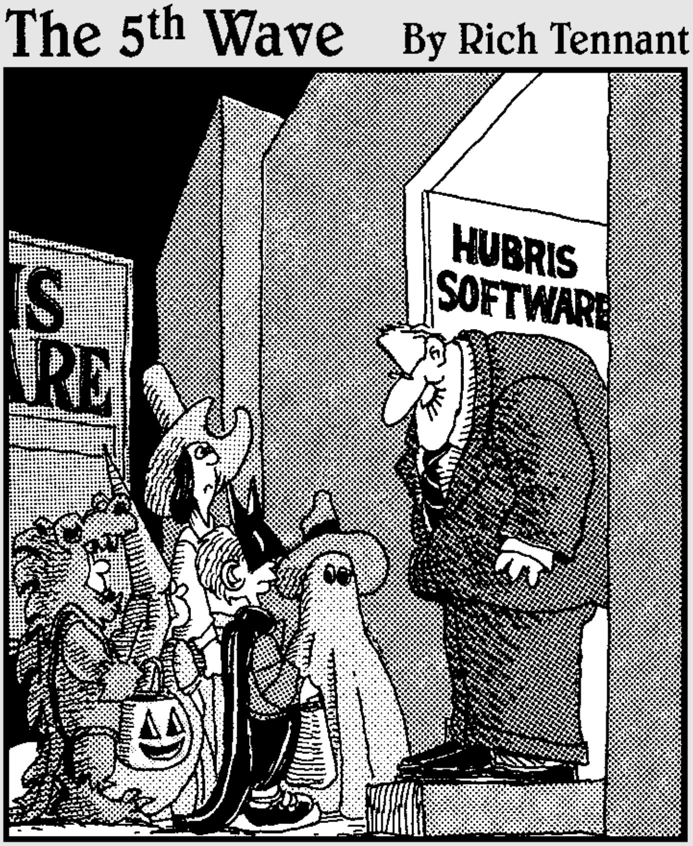
Category
All

Qualified BizTalk Server Architects and Developers wanted. Help build Mission Critical Solutions and connect clients with results!

Figure 4-5: Configure your application profile on Facebook.

Book VII

Extending the Family



"We've failed to meet our October 31 launch date for TREAT this year, but we're developing a new more robust version that should be available by the first or second quarter of next year."

Contents at a Glance

Chapter 1: Exploring Visual Studio Extensions

In This Chapter

- **Getting the most out of Visual Studio**
- **Building the next generation of Web sites with AJAX**
- **Looking ahead to the future of Visual Studio**

The name of this chapter should be "Getting Everything You Can from Visual Studio 2010, Now and in the Future." Since the release of Visual Studio 2008 in October 2007, Microsoft has released a ton of free add-ons that you can download from the Web.

Extending Visual Studio

You have many opportunities to extend Visual Studio, whether it be by using Microsoft add-ons or cool toys created by the community. This section lists a few of both and shares additional resources with you, including how to get hold of trial versions of development tools.

Adding administration and troubleshooting tools

One of the best things about add-ons is that all you have to do to get them is download them. Some help you administer and troubleshoot development projects; others provide new tools for programming Vista and Office 2007.

Administrative and troubleshooting downloads available from Microsoft include the following:

- **Team Foundation Server (TFS) Admin Tool:** As an administrator, add users to TFS through a single user interface (UI). Download the tool from

  ```
  http://msdn2.microsoft.com/tfs2008
  ```

- **Web Application Installer Projects:** Create a project to deploy your ASP.NET Web sites using a Windows Installer Package. Download from

  ```
  http://wai.codeplex.com/
  ```

- **CSS Control Adapter Toolkit:** Take control of the elements rendered by Web server controls, using Cascading Style Sheets (CSS). The toolkit lets you use CSS to override many server controls' default behavior of using `<table>` tags to render themselves to the browser. Download the toolkit from

  `www.asp.net/cssadapters`

  TIP: If you're not that familiar with using CSS for Web-page layout, pick up a copy of *CSS Web Design For Dummies,* by Richard Mansfield (Wiley). The book goes way beyond your basic CSS styling and shows you how to create multicolumn Web pages without using HTML tables.

- **Developer Highway Code:** Access and use an e-book that features guidance and security checklists for .NET 1.1, 2.0, 3.5, and 4.0. This PDF file covers writing secure code, brought to you by the Microsoft Patterns & Practices group. Download the e-book from

  `www.microsoft.com/uk/msdn/security/dev_highway.mspx`

- **Spec# Programming System:** Use the Spec# programming language to extend design-by-contract features to C#. Get more information at

  `http://research.microsoft.com/specsharp`

TECHNICAL STUFF: The Team Foundation Server Admin Tool is part of the Visual Studio Power Toys, which are released under the Microsoft Shared Source initiative. You can read more about Shared Source and find additional downloads at

`www.codeplex.com/PackInstaller`

TIP: CodePlex is a Web site that Microsoft dubs a "new collaborative development portal." CodePlex was formally launched in June 2006 with 30 open source projects for developers using Microsoft platforms. Be sure to check out the CodePlex Web site at `www.codeplex.com`.

Downloading new programming tools for Vista and Office

With the release of Windows 7 and Office, Microsoft made available several tools that target these newer programming models. Here are a few resources you should check out:

- **Extensions for Windows Workflow Foundation (WWF):** Provide support for building applications that use WWF
- **WinFX Development Tools:** Support building applications that target WinFX with Extensible Application Markup Language (XAML) support and project templates

- **.NET Framework version 4.0:** Provides the libraries you need to target all the features of Windows Vista and Windows 7
- **Windows Software Development Kit:** Includes documentation and samples that demonstrate the features of Windows Vista and Windows 7

You can find the latest version of these resources on the Windows Web site at

```
http://www.microsoft.com/windows/downloads/default.aspx
```

Many resources for programming Office 2007 are available at the Office Developer Center on MSDN. Some of these resources work with Visual Studio Professional 2010, and others require Visual Studio Tools for Office. Visit the Tools and Technologies page at

```
http://msdn.microsoft.com/en-us/vsto/default.aspx
```

Trying out development resources and new server products

Microsoft has a Web site where you can order several resources on DVD. The resources are free, although you do pay shipping and handling. The following resources are available for order at

```
http://msdn.microsoft.com/en-us/evalcenter/default.aspx
```

- **Team Foundation Server 180-day trial:** Includes 180-day trial of SQL Server 2008 Enterprise Edition and Team Foundation Server.
- **SQL Server 2008 Enterprise Edition 180-day trial:** Includes all of the features of SQL Server 2008 for you to evaluate.
- **Microsoft Developer Security Resource Kit:** Includes articles, whitepapers, Webcasts, and code samples.
- **Microsoft Patterns & Practices (January 2006 release):** Includes several patterns and practices guides, as well as the complete January 2006 release of the Enterprise Library.
- **Windows Mobile 5.0 Developer Resource Kit:** Includes resources to get started developing applications that target Microsoft's mobile platform.

The kits are guided tours of the resources available for a given topic, conveniently available on DVD. In the meantime, you can download Software Development Kits for Windows SharePoint Services version 3.0 at

```
www.microsoft.com/downloads/details.aspx?familyid=
    05E0DD12-8394-402B-8936-A07FE8AFAFFD&displaylang=en
```

You can find the SharePoint Server 2007 SDK at

```
www.microsoft.com/downloads/details.aspx?familyid=
    6D94E307-67D9-41AC-B2D6-0074D6286FA9&displaylang=en
```

Internet Information Services (IIS) 7 — the version of IIS — has integrated support for ASP.NET. IIS7 has its own dedicated Web site at `www.iis.net`, where you'll find technical resources, articles, blogs, and other community resources. Write your own modules — using the .NET Framework to target the IIS application programming interface — or become more knowledgeable about how to configure and tweak IIS. The Downloads section features an IIS7 Managed Module Starter Kit for C# or C++.

Other fun items you may want to download from the community include

- **Spell checker for ASP.NET:** Spell-checks the text you add to your ASP.NET Web pages. This was written by one of the developers on the Microsoft Web Development Tools Team in his spare time. In other words, the tool isn't supported by Microsoft. You can download the spell checker at

  `http://blogs.msdn.com/mikhailarkhipov`

- **NDepend:** Code analyzer that visually displays your source code's architecture and allows you to apply many code metrics against your source code. Download for free at

  `www.ndepend.com`

- **MSBuild Community Tasks Project:** An open source project that's building a library of common MSBuild tasks. You'll find just about everything you need to automate your nightly builds. Visit the project's Web site at

  `http://msbuildtasks.tigris.org`

- **RSS Toolkit:** Created by a member of the ASP.NET development team for consuming and publishing RSS feeds. Download the toolkit from

  `http://blogs.msdn.com/dmitryr/archive/2006/03/26/561200.aspx`

Keeping up with developer blogs and other news sources

Microsoft has embraced blogging. A *blog* (short for *Web log*) is a Web site where a blogger posts entries and visitors post comments on the blog entries. Here's a short list of blogs you may want to keep tabs on:

- **Web Development Tools Team Blog:** These are the folks that develop Visual Web Developer. Read their blog at

  `http://blogs.msdn.com/webdevtools`

- **S. “Soma” Somasegar’s blog:** Soma is the corporate vice president for the Microsoft Developer Division. His blog always features the latest news about Visual Studio 2010. Find his blog on MSDN at

  ```
  http://blogs.msdn.com/somasegar
  ```

- **Developer Division Customer Product Lifecycle Experience Team (DDCPX) Team Blog:** These are the folks responsible for bringing aftermarket solutions — such as the Visual Studio Power Toys and software development kits (SDKs) — to you. You’ll need a login to access this blog. Sign in at

  ```
  http://blogs.msdn.com
  ```

- **MSBuild Team Blog:** MSBuild is the build engine for Visual Studio. Read the team blog at

  ```
  http://blogs.msdn.com/msbuild
  ```

You can find more Microsoft blogs at

```
http://www.microsoft.com/communities/blogs/PortalHome.mspx
```

MSDN has created several developer centers that provide links to blogs, downloads, articles, and other resources of interest to developers. Some centers you may want to visit include

- **Security Developer Center:** `http://msdn.microsoft.com/security`
- **Visual Studio Developer Center:** `http://msdn.microsoft.com/vstudio`
- **Smart Client Developer Center:** `http://msdn.microsoft.com/smartclient`
- **Office Developer Center:** `http://msdn.microsoft.com/office`

You can find even more developer centers on MSDN at `http://msdn.microsoft.com/developercenters`.

Exploring AJAX and the Atlas Library

AJAX — Asynchronous JavaScript And XML — is a technology that allows developers to avoid traditional ASP.NET postbacks, in many cases, and provides a richer user experience with less delay. Developers use a combination of client-side JavaScript and server-side programming to create a Windows-like user experience. AJAX offers an important benefit: The browser and the server can communicate without a postback. From an end-user standpoint, that means no screen flash while the page refreshes. Popular Web sites using AJAX include

- Google Suggest
- Google Maps
- Flickr
- Gmail
- Outlook Web Access
- Yahoo! Mail Beta

To get a feel for some of the things you can do with AJAX, go to the demos page at `http://openrico.org`.

To get a feel for the underlying technologies of AJAX and what it takes to AJAX-ify your Web sites using technologies besides .NET, pick up a copy of *Ajax For Dummies,* by Steve Holzner (Wiley).

The technologies used in AJAX are nothing new. They've been around for a long time; only recently have folks applied the name AJAX to this style of programming. The technology that underlies AJAX is the object `XMLHttpRequest`. This one object makes it possible to send data back and forth between the client and server without a page refresh.

Atlas is Microsoft's version of AJAX. It consists of

- **Client-side library:** Atlas provides object-oriented access to the client-side features you would usually have to write in JavaScript. With Atlas, you can apply what you know about object-oriented programming in .NET without diving into JavaScript.
- **Server controls:** Server-based controls similar to existing Web server controls, such as buttons and text boxes.

Atlas makes extensive use of Web services to provide the server-side features of the client-server interaction of AJAX. It's quite common for developers to wire up the elements of their user interfaces to Web services that provide data (such as specific items for a drop-down list). Of course, you aren't limited to consuming only Web services that you create. You can also use an Atlas-enabled Web page to connect to third-party Web services.

Web sites that use content from other sources are called *mashups.* Mashups often utilize several technologies, including Web services, RSS, and AJAX. Mashups are a driving force in the *Web 2.0* movement. (The term often used to refer to the next generation of Web applications.)

Because the AJAX Toolkit is built into Visual Studio 2010, it no longer needs to be installed separately, like it did with Visual Studio 2005.

Looking Ahead to the Next Visual Studio

My guess is that this next-generation approach to development will allow developers to immerse themselves in modeling their business problem — and in generating code from that model — rather than getting stuck in a "code zone." Three development resources for a closer look are

- **The Microsoft Center for Software Excellence** (`www.microsoft.com/Windows/CSE`): A group in Microsoft Research focused on developer productivity at Microsoft.
- **Intentional Software** (`www.intentionalsoftware.com`): The company founded by Charles Simonyi to bring to market the next generation of software development. Simonyi is the man behind Microsoft Word and Excel. He also created the Hungarian notation-naming convention.
- **Software Factories** (`www.softwarefactories.com`): The idea of being able to mass-produce code from a model is being used now in Visual Studio Team System. Expect to see this concept trickle down to the individual developer in the future.

You can see how Visual Studio Team System is already using models to generate code in the next chapter of Book VII.

Chapter 2: Being a Team Player with Visual Studio Team System

In This Chapter

- **Figuring out the products in Visual Team System**
- **Exploring the tools of the new role-based editions**
- **Using Team Foundation Server**

If you've ever worked on a team project, you've probably experienced the frustrations of trying to collaborate via e-mail and folder shares. Invariably, an important document gets overwritten or an important stakeholder is left off the distribution list. Either way, the project suffers.

For most teams, the thought of implementing software to help with collaboration is about as much fun as having a root canal. Even if you decide on a vendor and manage to get the budget approved, you know the software's benefits are overshadowed by its daunting complexity.

The Microsoft approach to solving this problem is a bit different than most other vendors' approaches. Instead of creating an entire new set of tools for people to learn, Microsoft decided to create a solution that allows team members to keep using the same tools they've always used. The project managers keep using Excel and Project. The developers use Visual Studio. The architects get modeling tools that actually synchronize with code, and the testers finally get real software. The name for this make-everybody-happy approach to supporting team development is Visual Studio Team System.

Introducing the Visual Studio Team System

Visual Studio Team System (VSTS) isn't an edition of Visual Studio. Rather, it's a platform for building software in a team environment. The platform is composed of the following product offerings:

- **Four role-based vertical feature sets within Visual Studio:** Each feature set has a different set of tools tailored to a role in the software development life cycle. Visual Studio has specific features for the following roles on a team
  - Architect
  - Database professional

  - Developer
  - Tester

- **Visual Studio 2010 Ultimate:** This product suite includes all four role-based vertical feature sets.
- **Team Foundation Server (TFS) Standard Edition:** This server product's features enable the members of a team to collaborate and manage projects. All versions of Visual Studio 2010 — except the Express Edition — can access a Team Foundation Server.

  Each edition of Visual Studio in the VSTS comes with a client access license (CAL) that allows access to TFS. You must purchase a CAL separately for Visual Studio Standard or Professional editions. Each user accessing TFS must have a CAL.

  You aren't required to use TFS. You can use any of the Visual Studio editions without TFS.
- **Team Foundation Server Workgroup Edition:** This product provides the same features as TFS but is limited to five connections. TFS Workgroup Edition is included with any of the role-based versions of Visual Studio or the Team Suite.

If you have five or fewer members on your team, TFS Workgroup Edition is the way to go. You can upgrade to Standard Edition any time.

Each product in Visual Studio Team System comes with an MSDN Premium Subscription. Table 2-1 lists many of the tools you'll find in VSTS. (Descriptions for many of these tools appear later in the chapter.)

Table 2-1 Tools and Roles

Tool	*Roles*
Application Designer	Architect
Logical Infrastructure Designer	Architect
Deployment Designer	Architect
Class Designer	Architect, Developer
Visio and UML Modeling	Architect, Developer
Team Foundation Client	Architect, Developer
Visual Studio Professional	Architect, Developer
Dynamic Code Analyzer	Developer
Static Code Analyzer	Developer
Code Profiler	Developer
Unit Testing	Developer, Test

Tool	*Roles*
Code Coverage	Developer, Test
Load Testing	Test
Manual Testing	Test
Test Case Management	Test
Build Automation	Team Foundation Server
Change Management	Team Foundation Server
Work Item Tracking	Team Foundation Server
Reporting	Team Foundation Server
Project Site	Team Foundation Server
Integration Services	Team Foundation Server
Project Management	Team Foundation Server

Implementing VSTS

Visual Studio Team System is more than a new suite of tools: It supports the entire application life cycle and enables processes to improve continuously.

Still, implementing VSTS isn't a light-hearted undertaking. Visual Studio Team System integrates with several Microsoft server and client products. As a result, VSTS requires a *big* honkin' server with at least 15GB of storage just to get started, plus about 8GB of storage for each project. The server products required by VSTS include

- Windows Server 2003 R2 or Windows Server 2008
- SQL Server 2005 (Standard or Enterprise Edition) or SQL Server 2008 (Standard or Enterprise Edition)
- Windows SharePoint Services (WSS)Compared with other products of this type, VSTS is relatively simple to implement. A single server can support several hundred developers.

If you'd rather not deal with implementing yet another server, you can find a service provider to host a TFS for you. Developers connect to the hosted server as if it were sitting in your server room. You get all the productivity of TFS without any of the hassles of implementing and administering a server.

You can use a number of clients to access TFS. These include

- Visual Studio 2010
- Microsoft Office Excel 2003 or later

- Microsoft Project 2003 or later
- Internet Explorer

All the clients except Internet Explorer require the installation of Team Explorer (the Team Foundation client). See the section "Accessing Team Foundation Server," later in this chapter, for more about Team Explorer.

To take TFS for a test spin, you can download a 90-day trial edition from Microsoft's Web site at

```
www.microsoft.com/downloads/details.aspx?FamilyID=
    b0155166-b0a3-436e-ac95-37d7e39a440c
```

Discovering more about VSTS

To find out more about VSTS, check out these resources:

- **Team System Rocks Web site:** A Web site dedicated to VSTS that features blogs, forums, and tutorials. The site has dozens of videos on using VSTS. Visit the site at

  ```
  http://teamsystemrocks.com
  ```

- **Visual Studio Team System Developer Center:** View the VSTS portal on MSDN at

  ```
  http://msdn.microsoft.com/vstudio/teamsystem
  ```

- **Visual Studio Team System Virtual Labs:** Access tutorials hosted on MSDN at

  ```
  http://msdn.microsoft.com/en-us/aa570323.aspx
  ```

Exploring the New Role-Based Features of Visual Studio 2010 Ultimate

The role-based editions of Visual Studio have several tools. Some of these tools include the Application Designer in the Architect Edition. Other tools (such as static code analysis in the Developer Edition) simply integrate existing tools that many developers have used for a while. Either way, the new editions are long on features and compatibility.

Before you decide to spring for Visual Studio 2010 Ultimate, consider that you can achieve the same effect as many of these editions by using third-party tools. Depending on your situation, you may come out ahead by sticking with Visual Studio 2010 Professional and using a set of third-party tools such as these:

- **Unit Testing:** NUnit and NMock.
- **Modeling:** Visio for Enterprise Architects and Class Designer.
- **Build automation:** MSBuild and NAnt.
- **Static code analysis:** FXCop analyzes your code for compliance to standards.
- **Project portal:** Use WSS or create your own portal with ASP.NET 4 portal pages.

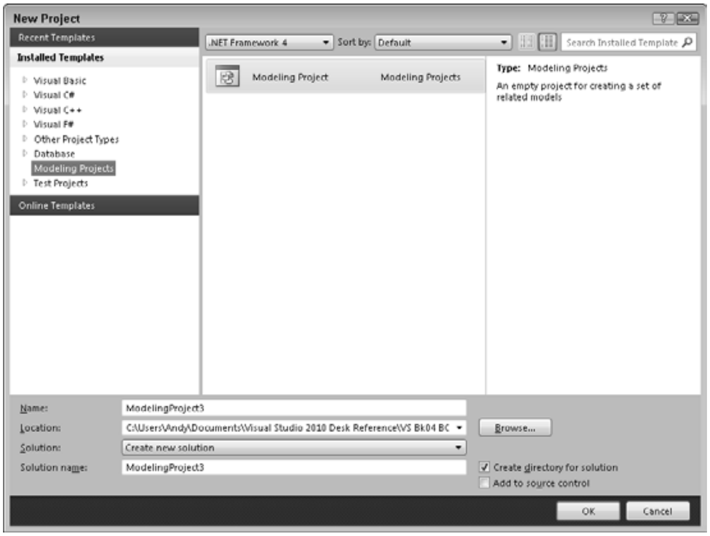
  Windows SharePoint Services is a component in Windows Server 2003 and Windows Server 2008. You can download WSS free from Microsoft at

  ```
  www.microsoft.com/downloads/details.aspx?FamilyId=
      B922B28D-806A-427B-A4C5-AB0F1AA0F7F9
  ```
- **Team Foundation Server:** Access TFS, using the Team Explorer add-in for Visual Studio 2008.

Visual Studio Ultimate for software architects

Visual Studio 2010 Ultimate includes a set of designers that allow architects to model their application designs. Visual Studio 2010 Ultimate ships with several project templates that allow you to create Unified Modeling Language (UML) diagrams.

To create UML diagrams in Visual Studio 2010 Ultimate, choose New➪Project, select Modeling Project from the list of project templates, and then click OK, as shown in Figure 2-1.



.NET Framework 4 Sort by: Default

Type: Modeling Projects

An empty project for creating a set of related models

Name: ModelingProject3

Location: C:\Users\Andy\Documents\Visual Studio 2010 Desk Reference\VS Bk04 BC

Solution: Create new solution

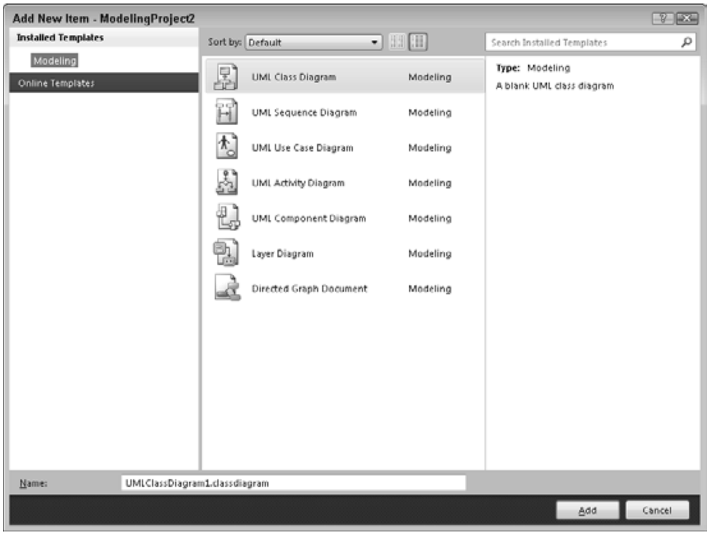
Solution name: ModelingProject3

☑ Create directory for solution

□ Add to source control

Figure 2-1: Modeling project templates.

Visual Studio 2010 Ultimate offers several types of UML diagrams to add to your modeling projects. To add a new UML diagram, choose Project➪Add New Item from the menu or Add New Item from the context menu on the project name in the Solution Explorer. Figure 2-2 shows the UML diagram templates.


Add New Item - ModelingProject2
Installed Templates
Modeling
Online Templates
Sort by: Default
Search Installed Templates
UML Class Diagram Modeling
UML Sequence Diagram Modeling
UML Use Case Diagram Modeling
UML Activity Diagram Modeling
UML Component Diagram Modeling
Layer Diagram Modeling
Directed Graph Document Modeling
Add
Cancel

Type: Modeling
A blank UML class diagram

Name: UMLClassDiagram1.classdiagram

Figure 2-2: Diagrams to add to your modeling project.

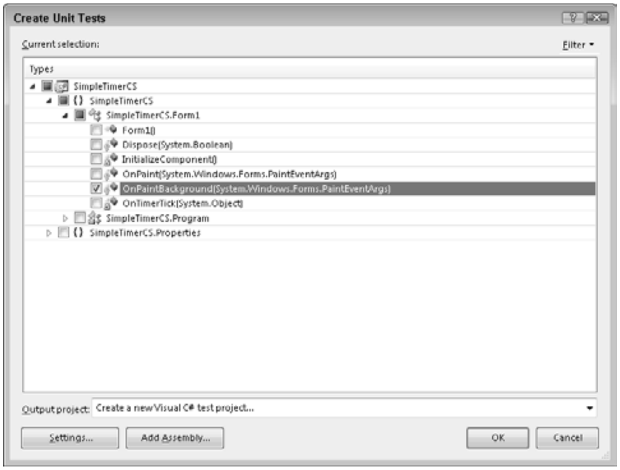
Visual Studio 2010 Ultimate for software developers

Visual Studio 2010 Ultimate provides additional tools that developers can use to write better code. Many of these tools are available elsewhere as free downloads or open-source software.

Visual Studio Team System for Software Developers integrates the following tools into the development environment:

- **Static code analysis:** This tool uses rules and patterns to detect errors and bad coding form.
- **Dynamic code analysis:** This code-profiler monitors an application’s performance while it’s running.
- **Unit testing:** This framework lets you generate and manage unit tests.
- **Code coverage:** This tool provides feedback on how effective your unit tests are at covering all your code.

Developers generate unit tests from the Code Editor, as shown in Figure 2-3.


Create Unit Tests

Current selection:

Types

- ■ SimpleTimerCS
  - ■ { } SimpleTimerCS
    - ■ SimpleTimerCS.Form1
      - ☐ Form1()
      - ☐ Dispose(System.Boolean)
      - ☐ InitializeComponent()
      - ☐ OnPaint(System.Windows.Forms.PaintEventArgs)
      - ☑ OnPaintBackground(System.Windows.Forms.PaintEventArgs)
      - ☐ OnTimerTick(System.Object)
    - ☐ SimpleTimerCS.Program
  - ☐ { } SimpleTimerCS.Properties

Output project: Create a new Visual C# test project...

Figure 2-3: Some testing functions are built into the developer version.

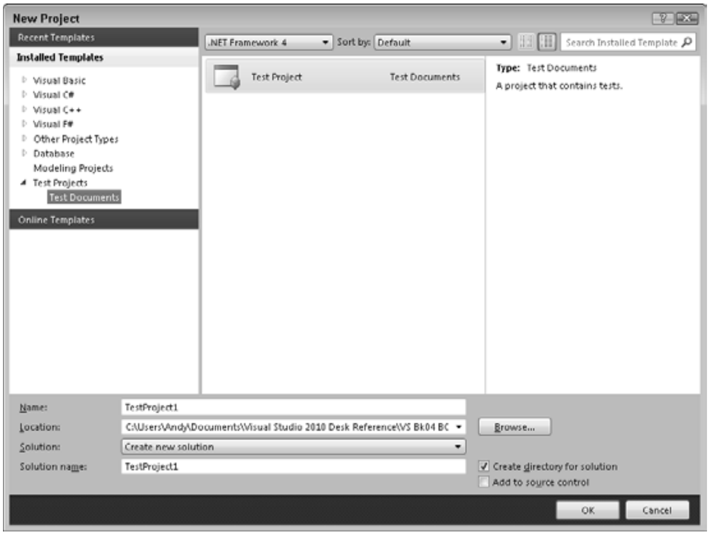
Visual Studio 2010 Ultimate for software testers

Visual Studio 2010 for software testers provides resources for creating and managing many kinds of tests. You can use the Tester Edition to

- Create unit tests.
- Record Web tests.
- Execute a sequential list of tests.
- Measure your application's performance by running any combination of unit tests and Web tests over and over again.
- Capture the steps for manual tests.
- Turn any code into a test.

You organize your tests into projects by using a Test Project template in Visual Studio, as shown in Figure 2-4.

A Test Run Configuration is added by default to the solution when a Test Project is added. Use the Test Run Configuration to configure the testing environment and code coverage, as shown in Figure 2-5.



.NET Framework 4

Sort by: Default

Test Project   Test Documents

Type: Test Documents

A project that contains tests.

Name: TestProject1

Location: C:\Users\Andy\Documents\Visual Studio 2010 Desk Reference\VS Bk04 BC

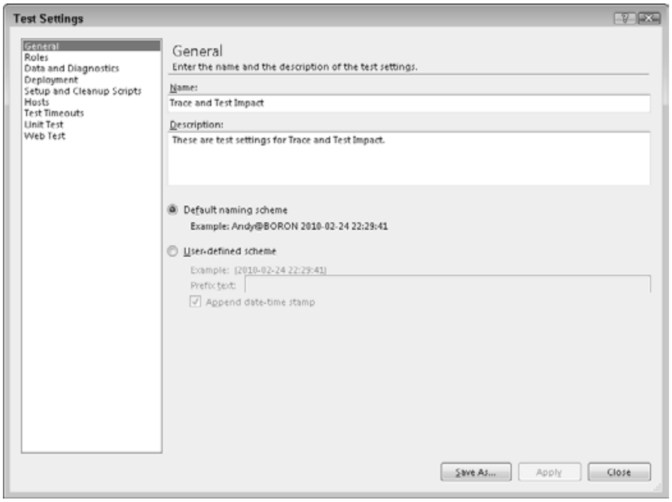
Solution: Create new solution

Solution name: TestProject1

☑ Create directory for solution

□ Add to source control

Figure 2-4: Testing is much easier with the Visual Studio Ultimate 2010 for testers.



General

Enter the name and the description of the test settings.

Name: Trace and Test Impact

Description: These are test settings for Trace and Test Impact.

● Default naming scheme

Example: Andy@BORON 2010-02-24 22:29:41

○ User-defined scheme

Example: (2010-02-24 22:29:41)

Prefix text:

☑ Append date-time stamp

Figure 2-5: The Test Run Configuration is added by default for your convenience.

You add a Test Project from the New Project window, just like you would add any project in Visual Studio.

Visual Studio 2010 Ultimate for database professionals

Visual Studio 2010 Ultimate brings the database developer closer into the software-development loop. With Visual Studio 2010 Ultimate, you can do the following:

- Refactor database schema objects.
- Automate testing with unit tests.
- Place database schemas under source control.
- Use `SchemaCompare` and `DataCompare` to compare two databases and generate scripts to synchronize the databases.
- Populate test databases based on production data, using the `DataGenerator`.

Getting to Know Team Foundation Server

Team Foundation Server — the bedrock of collaboration features in VSTS — provides the following:

- **Source control management:** Create check-in policies that define the actions your developers must complete before integrating source code.
- **Work item tracking:** Create tasks, assign tasks to team members, and associate tasks with actions, such as code check-in.
- **Build automation:** Step through a wizard to create complex automated build processes.
- **Reporting:** Generate reports on all activities in your TFS.
- **Centralized project site:** Set up a project site where all stakeholders can access project artifacts.
- **Project management:** Perform project management tasks in familiar tools, such as Excel and Project.

Many of the features in TFS are possible, using the technologies of WSS. WSS version 3 is available as a free download from the Microsoft Web site. Before you take the plunge with VSTS, install WSS. Also, pick up a copy of *Microsoft SharePoint 2007 For Dummies,* by Vanessa Williams (Wiley). You'll find scenarios that show you how to use SharePoint in many different business contexts.

Accessing Team Foundation Server

You must install the Team Explorer client before you can access TFS. Team Explorer is an add-in for Visual Studio. You can install the client using the Team Foundation Server software or download the add-in from Microsoft Downloads. You don't need to have any edition of Visual Studio to use Team Explorer.

To download the client from Microsoft, access this link:

```
www.microsoft.com/downloads/details.aspx?familyid=
   0ED12659-3D41-4420-BBB0-A46E51BFCA86&displaylang=en
```

To use Team Explorer to access an existing project on a TFS, follow these steps:

1. **Choose View➪Team Explorer.**

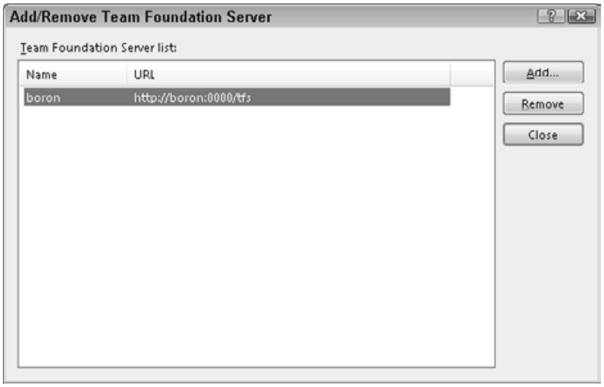
   The Team Explorer window appears.

2. **Click the Add Existing Team Project button.**

   The Connect to a Team Foundation Server window appears.

3. **Click the Servers button.**

   The Add/Remove Team Foundation Server window appears (see Figure 2-6). Alternatively, you can select an existing server from the Team Foundation Server drop-down list.



Add/Remove Team Foundation Server

Team Foundation Server list:

Name	URL
boron	http://boron:0000/tfs

Figure 2-6: You can easily connect to a local server.

4. **Click Add.**

   The Add Team Foundation Server appears.

5. **Type the server's name in the Name text box.**

If your administrator instructs you to use a different port or protocol, enter those in the Connection Details section. Figure 2-6 shows a connection to a server. This shows the default port. In some cases, this port will conflict with other third-party products. For example, Visual MainWin also relies on this port. To ensure that there isn't any conflict, you need to change the port used by one of the two products.

6. **Click OK.**
7. **If prompted, type your username and password.**

   The Connect to a Team Project dialog box appears.
8. **Select a project from the list of projects, as shown in Figure 2-7.**

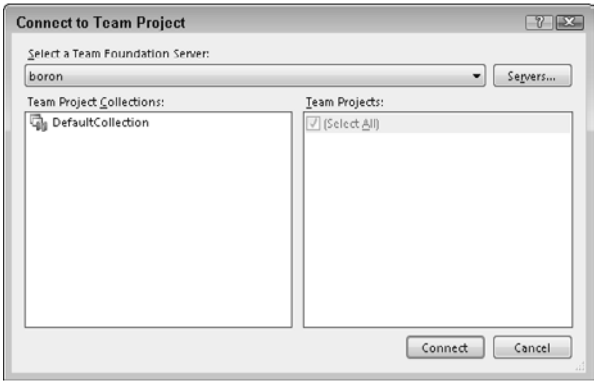


Figure 2-7: You'll see a list of projects once you provide your username and password.

9. **Click OK.**

   The project appears in the Team Explorer.

Creating a new team project

You may wonder why Microsoft didn't create a Visual Studio 2010 Ultimate features for project managers. It's probably because project managers don't use Visual Studio that often. And the Ultimate edition accommodates their style of work by interfacing with Excel and Project.

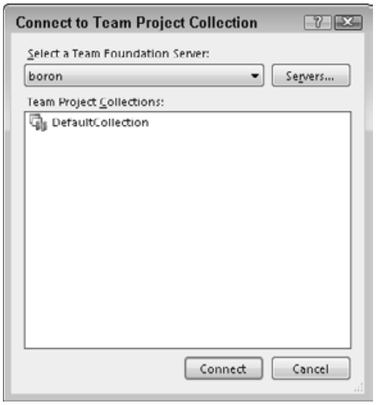
Some tasks, such as creating new team projects, must be completed by using Visual Studio. Visual Studio provides a group of predefined environment settings specific to project managers. To reset your settings to use the Project Management Settings, use the Import and Export Settings Wizard, as described in Book II, Chapter 1. The project manager environment settings eliminate most of the menu items used by developers and other roles.

The New Team Project Wizard walks you through the process of creating a new team project. To start the wizard, follow these steps:

1. **Choose File➪New Team Project.**

   The New Team Project Wizard appears.

   Alternatively, you can choose File➪New➪Team Project. If you're not connected to a team server, the Connect to Team Project Collection dialog box appears, as shown in Figure 2-8.



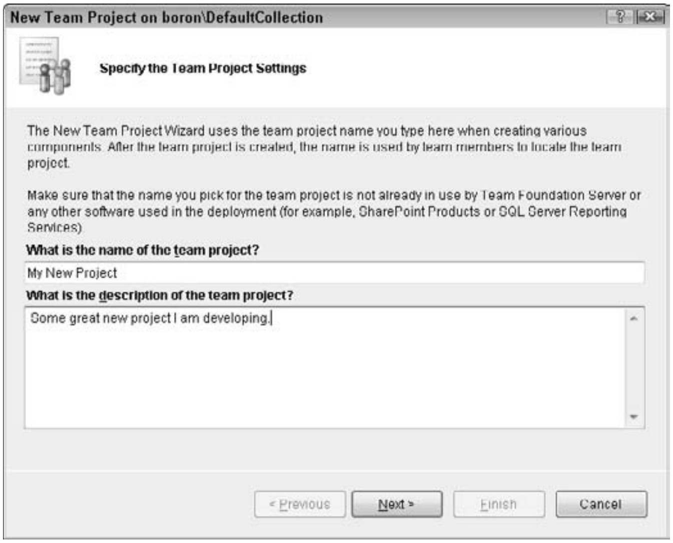
Select a Team Foundation Server: boron

Team Project Collections: DefaultCollection

Figure 2-8: Connect to a Team Project Collection.

2. **Click Connect.**

   The New Team Project wizard appears, as shown in Figure 2-9.


New Team Project on boron\DefaultCollection
Specify the Team Project Settings
< Previous
Next >
Finish
Cancel

The New Team Project Wizard uses the team project name you type here when creating various components. After the team project is created, the name is used by team members to locate the team project.

Make sure that the name you pick for the team project is not already in use by Team Foundation Server or any other software used in the deployment (for example, SharePoint Products or SQL Server Reporting Services).

What is the name of the team project?

My New Project

What is the description of the team project?

Some great new project I am developing.

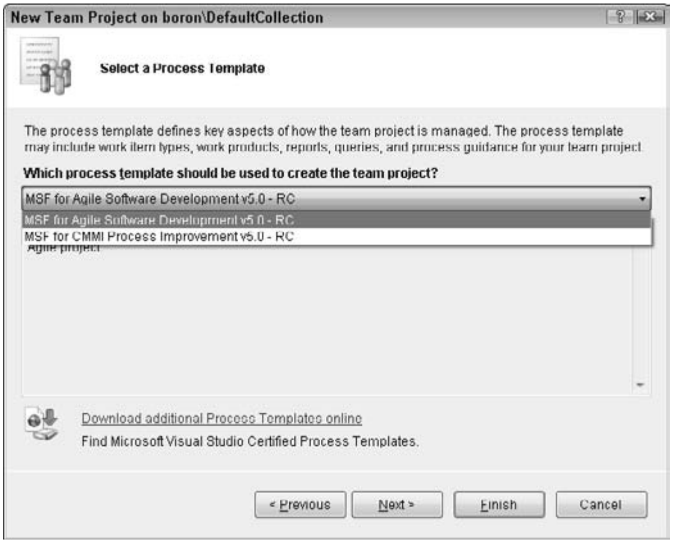
Figure 2-9: Give your project a name and description.

3. **Enter a name and description for your project and then click Next.**

   The project name must be unique on the Team Foundation Server.

   The Select a Process Template step appears. The *process template* defines workflow for your project. Team Foundation Server includes two process templates from the Microsoft Solutions Framework; you can extend them or create your own.

4. **Select a process template from the drop-down list and click Next, as shown in Figure 2-10.**


New Team Project on boron\DefaultCollection
Select a Process Template
Download additional Process Templates online
< Previous
Next >
Finish
Cancel

The process template defines key aspects of how the team project is managed. The process template may include work item types, work products, reports, queries, and process guidance for your team project.

Which process template should be used to create the team project?

MSF for Agile Software Development v5.0 - RC

MSF for Agile Software Development v5.0 - RC
MSF for CMMI Process Improvement v5.0 - RC

Agile project

Find Microsoft Visual Studio Certified Process Templates.

Figure 2-10: You must select a process template from the drop-down list.

The steps that appear next depend upon the process template you select. Steps 5 through 9 in this list are based on the Microsoft Solutions Framework (MSF) for the Agile Software Development process template.

Visit the VSTS Developer Center on MSDN to get more information on using development processes from the Microsoft Solutions Framework at

```
http://msdn.microsoft.com/vstudio/teamsystem/msf
```

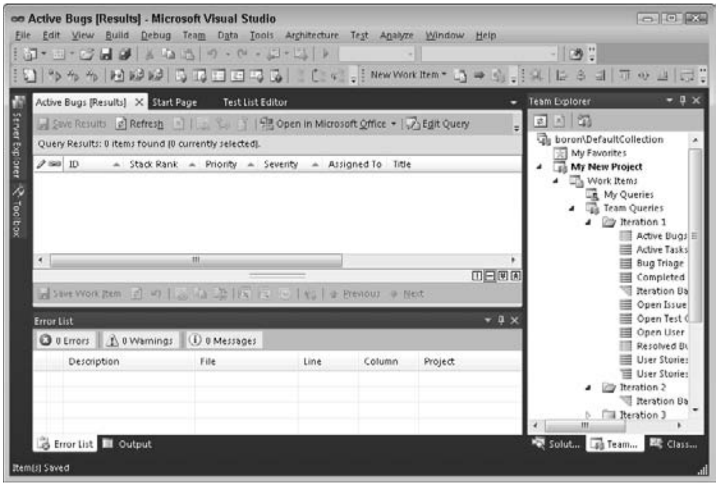
The Specify Source Control Settings window appears.

5. **Select your settings to establish source control for the project then click Next.**

   The Confirmation Page appears.

6. **Review your settings and click Finish.**

   The wizard creates your project. The project is now accessible from the Team Explorer (see Figure 2-11).



Query Results: 0 items found (0 currently selected).

Figure 2-11: The sample project portal.

Browsing with the Team Explorer

Team Explorer is the premier client for browsing resources in a Team Project. You open the Team Explorer from the View menu. The Team Explorer displays the following:

- **Work items:** Lists the work-item queries in your team project. Click a query to expand a list of work orders.
- **Documents:** Displays the document libraries from the project portal. The project's process template determines which document libraries are created.
- **Reports:** Displays a list of reports available for your projects. Reports are created by using SQL Server Reporting Services.
- **Team Builds:** Displays a list of build types for your team project.
- **Source control:** Defines check-in policies for the project.

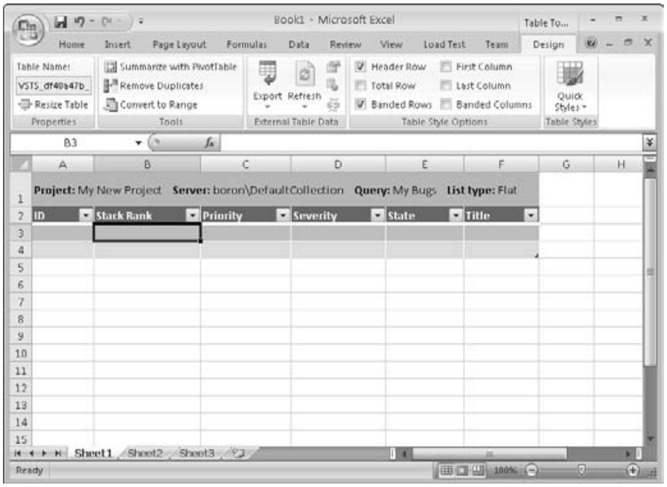
Figure 2-12 shows a work-item query accessed via Team Explorer.

Creating work items with Excel

You can connect to projects in a TFS by using Excel or Project. Team Explorer adds a Team toolbar that you can use to get work items and to publish new or modified items to the server. To connect to a new work-item list in Excel, follow these steps:

1. **Click the New List button on the Team toolbar.**

   The Connect to Team Foundation Server window appears. (Refer to Figure 2-8.)



Project: My New Project **Server:** boron\DefaultCollection **Query:** My Bugs **List type:** Flat

ID	Stack Rank	Priority	Severity	State	Title

Figure 2-12: Access the work-item query via Team Explorer.

2. **Connect to your server, as described in the preceding example.**
3. **Select a team project from the list of projects and then click OK.**

   The New List window appears.
4. **Select the Query List radio button to access a list of work items.**

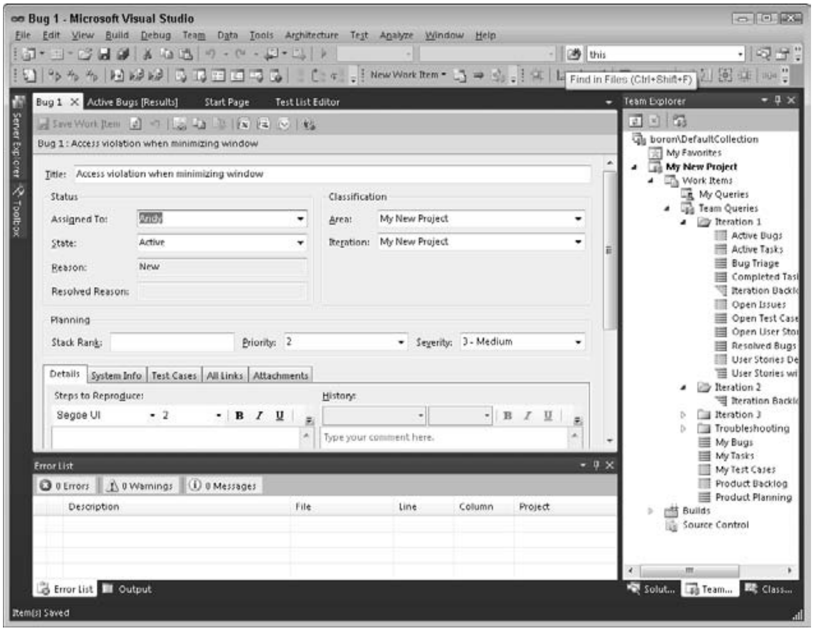
   Alternatively, click Input List to enter new work items and get individual work items.
5. **Click the Select a Query drop-down list.**

   A list of predefined work item queries appears.
6. **Select a query from the list and then click OK.**

   The query's results appear as a spreadsheet (see Figure 2-13).

The work items are retrieved into an Excel list. You can add new work items or update existing items. Click the Publish button in the toolbar to save your changes to the server, and your saved changes are immediately available to other team members.

You can add work items in Excel, Project, or Team Explorer; or, during source-control check-in.



Bug 1 - Microsoft Visual Studio

Bug 1 : Access violation when minimizing window

Title: Access violation when minimizing window

Status

Assigned To: Andy

State: Active

Reason: New

Resolved Reason:

Classification

Area: My New Project

Iteration: My New Project

Planning

Stack Rank:

Priority: 2

Severity: 3 - Medium

Item(s) Saved

Figure 2-13: You can select the query and see the results.

Index

Software Development & Programming/
Microsoft

It's new and better than ever! Learn to use Visual Studio 2010 with this book

Developers, listen up! Visual Studio 2010 offers many interface changes and a dizzying array of editors, controls, designers, and other tools. Whether you're starting from scratch with Visual Studio or need to boost your knowledge of coding, building apps, or accessing data, you'll find what you need to be productive.

- ***Get what you need*** — ***newbies can learn Visual Studio basics while more experienced developers can focus on what's new in the 2010 version***
- ***Code mode*** — ***create source code with the code editor, explore class anatomy in .NET, and debug, analyze, and test your code***
- ***Building blocks*** — ***explore types of applications you can build; learn to create C++, Windows, and Web applications; and deploy them on the cloud***
- ***Digging data*** — ***look at the new data access code-generation features in Visual Studio***
- ***Take control*** — ***manage the build process and apply source code control***

Open the book and find:

- **The languages of .NET**
- **Advice on installing Visual Studio**
- **Basic info about object-oriented programming**
- **Tips on handling exceptions and debugging**
- **How to lay out a Web site**
- **Why you need ADO.NET**
- **How to build applications for Facebook**
- **What the Visual Studio Team System offers**

Visit the companion Web site at www.dummies.com/go/vs2010, where you'll find a section featuring source code, resource links, and sample applications for each minibook.

Go to Dummies.com®
for videos, step-by-step examples, how-to articles, or to shop!

Andrew Moore is an experienced software developer with expertise in C, C++, and C#. Currently a senior software engineer, he has development experience with air traffic management, discrete-event simulation, and most recently with VoIP and multimedia business communication systems.

For Dummies®
A Branded Imprint of
WILEY

$39.99 US / $47.99 CN / £27.99 UK

ISBN 978-0-470-53943-9

53999

9 780470 539439